

8.0

*IBM MQ Tworzenie odwołania do
aplikacji*



Uwaga

Przed skorzystaniem z niniejszych informacji oraz produktu, którego one dotyczą, należy zapoznać się z informacjami zamieszczonymi w sekcji [“Uwagi” na stronie 2089](#).

To wydanie dotyczy wersji 8 wydania 0 produktu IBM® MQ oraz wszystkich kolejnych wydań i modyfikacji, o ile nie zostanie to określone inaczej w nowych edycjach.

Wysyłając informacje do IBM, użytkownik przyznaje IBM niewyłączne prawo do używania i rozpowszechniania informacji w dowolny sposób, jaki uzna za właściwy, bez żadnych zobowiązań wobec ich autora.

© Copyright International Business Machines Corporation 2007, 2023.

Spis treści

Tworzenie odwołania do aplikacji.....	7
Skorowidz aplikacji MQI.....	7
Przykłady kodu.....	8
Stale.....	61
Typy danych używane w interfejsie MQI.....	237
Wywołania funkcji.....	630
Atrybuty obiektów.....	810
Kody powrotu.....	885
Reguły sprawdzania poprawności opcji MQI.....	886
Komunikaty w kolejce publikowania/subskrypcji w kolejce.....	889
Kodowanie komputera.....	912
Opcje raportów i flagi komunikatów.....	915
Konwersja danych.....	919
Właściwości określone jako elementy MQRFH2.....	942
konwersja stron kodowych.....	951
Standardy kodowania na platformach 64-bitowych.....	1005
Skorowidz programistyczny aplikacji IBM i (ILE/RPG).....	1009
Opisy typów danych.....	1009
Wywołania funkcji.....	1270
Atrybuty obiektów.....	1395
Aplikacje.....	1441
Kody powrotu dla IBM i (ILE RPG).....	1455
Reguły sprawdzania poprawności opcji MQI dla produktu IBM i (ILE RPG).....	1456
Kodowanie komputera w systemie IBM i.....	1459
Opcje raportów i flagi komunikatów.....	1462
Konwersja danych.....	1465
Przetwarzanie konwersji.....	1466
Konwencje przetwarzania.....	1467
Konwersja komunikatów raportu.....	1471
MQDXP-Dane-parametr wyjścia konwersji danych.....	1472
MQXCNCV-Przekształć znaki.....	1477
MQCONVX-wyjście konwersji danych.....	1482
SOAP-odwołanie.....	1486
amqwSOAPNETListener: program nasłuchujący SOAP IBM MQ dla środowiska .NET 1 lub 2.....	1486
amqswsdl : generowanie pliku WSDL dla usługi .NET Framework 1 lub 2.....	1489
amqwclientconfig : utwórz deskryptor wdrażania klienta usług Web Service Axis 1.4 dla transportu IBM MQ dla protokołu SOAP.....	1490
amqwdeployWMQService : wdrażanie programu narzędziowego usług Web Service.....	1490
amqwRegisterdotNet : zarejestruj transport IBM MQ dla protokołu SOAP do .NET.....	1498
Licencja na oprogramowanie Apache.....	1499
Ustawienia SOAP produktu MQMD	1502
Ustawienia SOAP produktu MQRFH2.....	1508
runivt : transport produktu IBM MQ do sprawdzania poprawności instalacji SOAP.....	1510
Bezpieczne usługi Web Service przy użyciu transportu IBM MQ dla protokołu SOAP.....	1512
SimpleJavaListener : program nasłuchujący SOAP IBM MQ dla osi 1.4.....	1517
Obiekty nasłuchiwanie SOAP produktu IBM MQ.....	1519
Transport produktu IBM MQ dla nadawcy SOAP.....	1524
Transakcje.....	1526
Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service.....	1526
Identyfikator URI W3C SOAP over JMS dla klienta IBM MQ Axis 2.....	1534
Obsługiwane usługi Web Service.....	1540
Transport produktu IBM MQ dla klientów usług Web Service SOAP.....	1543

Procedury zewnętrzne, wyjścia funkcji API i odwołania do usług instalowalnych.....	1545
Struktura MQIEP.....	1546
Dane wyjściowe wyjścia konwersji danych.....	1549
MQ_PUBLISH_EXIT-wyjście publikowania.....	1553
Wywołania wyjścia kanału i struktury danych.....	1561
Wywołania wyjścia obciążenia klastra i struktury danych.....	1626
Odwołanie do wyjścia funkcji API.....	1652
Informacje uzupełniające o interfejsie usług instalowalnych.....	1714
Informacje uzupełniające o interfejsie usług instalowalnych.....	1779
Materiał odniesienia dla IBM MQ bridge for HTTP.....	1819
HTTP DELETE : komenda IBM MQ bridge for HTTP.....	1819
HTTP GET : komenda IBM MQ bridge for HTTP.....	1823
HTTP POST : komenda IBM MQ bridge for HTTP.....	1826
Nagłówki HTTP.....	1829
Kody powrotu HTTP.....	1845
Typy komunikatów i odwzorowania komunikatów dla mostu WebSphere dla protokołu HTTP....	1853
Format identyfikatora URI.....	1855
Klasy i interfejsy IBM MQ .NET.....	1856
Klasa MQAsyncStatus.NET.....	1856
Klasa MQAuthenticationInformationRecord.NET.....	1857
Klasa MQDestination.NET.....	1858
Klasa MQEnvironment.NET.....	1861
Klasa MQException.NET.....	1863
Klasa MQGetMessageOptions.NET.....	1864
Klasa MQManagedObject.NET.....	1867
Klasa MQMessage.NET.....	1870
Klasa MQProcess.NET.....	1882
Klasa MQPropertyDescriptor.NET.....	1884
Klasa MQPutMessageOptions.NET.....	1886
Klasa MQQueue.NET.....	1889
Klasa MQQueueManager.NET.....	1897
Klasa MQSubscription.NET.....	1910
Klasa MQTopic.NET.....	1911
Interfejs IMQObjectTrigger.NET.....	1917
Interfejs MQC.NET.....	1918
Identyfikatory zestawów znaków dla aplikacji produktu .NET.....	1918
Klasy C++ w programie IBM MQ.....	1921
Skorowidz języka C++ i MQI.....	1922
ImqAuthentication-rejestrowanie klasy C++.....	1938
Klasa ImqBinary C++.....	1940
Klasa ImqCache C++.....	1942
Klasa języka C++ ImqChannel.....	1945
ImqCICSBridgeHeader C++.....	1950
Klasa języka C++ ImqDeadLetterHeader.....	1957
Klasa ImqDistribution-lista C++.....	1959
Klasa języka C++ ImqError.....	1961
ImqGetMessageOptions klasa C++.....	1962
Klasa języka C++ ImqHeader.....	1965
ImqIMSBridgeHeader C++.....	1967
Klasa ImqItem C++.....	1970
Klasa języka C++ ImqMessage.....	1971
Klasa ImqMessageTracker C++.....	1978
Klasa ImqNamelist C++.....	1981
Klasa języka C++ ImqObject.....	1983
Klasa ImqProcess C++.....	1989
Klasa języka C++ ImqPutMessageOptions.....	1990
Klasa ImqQueue C++.....	1992
Klasa C++ programu ImqQueueManager.....	2003

Klasa ImqReferencenagłówka C++.....	2019
Klasa ImqString C++.....	2022
Klasa ImqTrigger C++.....	2027
Klasa ImqWorknagłówka C++.....	2030
Właściwości obiektów IBM MQ classes for JMS.....	2032
Zależności między właściwościami obiektów produktu IBM MQ classes for JMS.....	2036
APPLICATIONNAME.....	2038
WYJĄTEK ASYNCEXCEPTION.....	2038
BROKERCCDURSUBQ.....	2039
BROKERCCSUBQ.....	2040
BROKERCONQ.....	2040
BROKERDURSUBQ.....	2041
BROKERPUBQ.....	2041
BROKERPUBQMGR.....	2042
BROKERQMGR.....	2042
BROKERSUBQ.....	2042
BROKERVER.....	2043
CCDTURL.....	2044
CCSID.....	2044
CHANNEL.....	2044
CLEANUP.....	2045
CLEANUPINT.....	2045
Lista CONNECTIONNAMELIST.....	2046
CLIENTRECONNECTOPTIONS.....	2046
CLIENTRECONNECTTIMEOUT.....	2047
CLIENTID.....	2048
CLONESUPP.....	2048
COMPHDR.....	2049
COMPMSG.....	2049
CONNOPT.....	2049
CONNTAG.....	2051
opis.....	2051
DIRECTAUTH.....	2051
ENCODING.....	2052
EXPIRY.....	2053
FAILIFQUIESCE.....	2053
HOSTNAME.....	2054
LOCALADDRESS.....	2055
MAPNAMESTYLE.....	2055
MAXBUFFSIZE.....	2056
MDREAD.....	2056
MDWRITE.....	2057
MDMSGCTX.....	2057
MSGBATCHSZ.....	2058
MSGBODY.....	2059
MSGRETENTION.....	2059
MSGSELECTION.....	2060
MULTICAST.....	2060
OPTIMISTICPUBLICATION.....	2061
OUTCOMENOTIFICATION.....	2062
PERSISTENCE.....	2062
POLLINGINT.....	2063
PORT.....	2063
PRIORYTET.....	2064
PROCESSDURATION.....	2064
PROVIDERVERSION.....	2065
PROXYHOSTNAME.....	2067
PROXYPORT.....	2068

PUBACKINT.....	2068
PUTASYNCALLOWED.....	2069
QMANAGER.....	2069
QUEUE.....	2070
READAHEADALLOWED.....	2070
READAHEADCLOSEPOLICY.....	2071
RECEIVECCSID.....	2071
RECEIVECONVERSION.....	2072
RECEIVEISOLATION.....	2072
RECEXIT.....	2073
RECEXITINIT.....	2073
REPLYTOSTYLE.....	2074
RESCANINT.....	2074
SECEXIT.....	2075
SECEXITINIT.....	2076
SENDCHECKCOUNT.....	2076
SENDEXIT.....	2076
SENDEXITINIT.....	2077
SHARECONVALLOWED.....	2078
SPARSESUBS.....	2078
SSLCIPHERSUITE.....	2079
SSLCRL.....	2079
SSLFIPSREQUIRED.....	2079
SSLPEERNAME.....	2080
SSLRESETCOUNT.....	2080
STATREFRESHINT.....	2081
SUBSTORE.....	2081
SYNCPOINTALLGETS.....	2082
TARGCLIENT.....	2082
TARGCLIENTMATCHING.....	2083
TEMPMODEL.....	2083
TEMPQPREFIX.....	2084
TEMPTOPICPREFIX.....	2084
TOPIC.....	2085
TRANSPORT.....	2085
WILDCARDFORMAT.....	2086
Właściwość ENCODING.....	2086
Właściwości SSL obiektów JMS.....	2087
Uwagi.....	2089
Informacje dotyczące interfejsu programistycznego.....	2090
Znaki towarowe.....	2091

Tworzenie odwołania do aplikacji

Odsyłacze zawarte w tej sekcji ułatwiają tworzenie aplikacji produktu IBM MQ .

- [“Skorowidz aplikacji MQI” na stronie 7](#)
- [“Procedury zewnętrzne, wyjścia funkcji API i odwołania do usług instalowalnych” na stronie 1545](#)
-  [“Skorowidz programistyczny aplikacji IBM i \(ILE/RPG\)” na stronie 1009](#)
- [“SOAP-odwołanie” na stronie 1486](#)
- [“Materiał odniesienia dla IBM MQ bridge for HTTP” na stronie 1819](#)
- [“Klasy i interfejsy IBM MQ .NET” na stronie 1856](#)
- [“Klasy C++ w programie IBM MQ” na stronie 1921](#)
- [Biblioteki produktu IBM MQ classes for Java™](#)
- [IBM MQ classes for JMS](#)

Informacje pokrewne

[Projektowanie aplikacji](#)

Skorowidz aplikacji MQI

Odsyłacze zawarte w tej sekcji ułatwiają programowanie aplikacji interfejsu kolejki komunikatów (Message Queue Interface-MQI).

- [“Przykłady kodu” na stronie 8](#)
- [“State” na stronie 61](#)
- [“Typy danych używane w interfejsie MQI” na stronie 237](#)
- [“Wywołania funkcji” na stronie 630](#)
- [“Atrybuty obiektów” na stronie 810](#)
- [“Kody powrotu” na stronie 885](#)
- [“Reguły sprawdzania poprawności opcji MQI” na stronie 886](#)
- [“Kodowanie komputera” na stronie 912](#)
- [“Opcje raportów i flagi komunikatów” na stronie 915](#)
- [“Konwersja danych” na stronie 919](#)
- [“Właściwości określone jako elementy MQRFH2” na stronie 942](#)
- [“konwersja stron kodowych” na stronie 951](#)

Pojęcia pokrewne

[“Procedury zewnętrzne, wyjścia funkcji API i odwołania do usług instalowalnych” na stronie 1545](#)

Odsyłacze zawarte w tej sekcji ułatwiają programowanie wyjść użytkownika, wyjść funkcji API i aplikacji usług instalowalnych:

Odsyłacze pokrewne

[“SOAP-odwołanie” na stronie 1486](#)

Transport IBM MQ dla informacji o odwołaniu SOAP ułożonych alfabetycznie.

[“Materiał odniesienia dla IBM MQ bridge for HTTP” na stronie 1819](#)

Tematy dotyczące produktu IBM MQ bridge for HTTP, uporządkowane alfabetycznie

[“Klasy i interfejsy IBM MQ .NET” na stronie 1856](#)

Klasy i interfejsy IBM MQ .NET są uporządkowane alfabetycznie. Opisywane są właściwości, metody i konstruktory.

[“Klasy C++ w programie IBM MQ” na stronie 1921](#)

Klasy języka C++ programu IBM MQ hermetyzują interfejs kolejki komunikatów produktu IBM MQ (MQI). Dostępny jest pojedynczy plik nagłówkowy C++ **imqi.hpp**, który obejmuje wszystkie te klasy.

Informacje pokrewne

Projektowanie aplikacji

Klasy IBM MQ dla bibliotek produktu Java

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

Przykłady kodu

Informacje uzupełniające zawarte w tej sekcji umożliwiają realizację zadań, które dotyczą potrzeb biznesowych użytkownika.

Przykłady języka C

Ta kolekcja tematów jest najczęściej pobierana z przykładowych aplikacji produktu IBM MQ for z/OS . Mają one zastosowanie do wszystkich platform, z wyjątkiem przypadków, gdy jest to zauważone.

Nawiąże połączenie z menedżerem kolejek

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

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BCA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

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

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

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

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

Rozłączenie z menedżerem kolejek

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

Zmienne używane w tym ekstrakcie kodu to te, które zostały ustawione w produkcie “Nawiąże połączenie z menedżerem kolejek” na stronie 8. Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BCA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu utworzenia kolejki dynamicznej.

Ten ekstrakt jest pobierana z przykładowej aplikacji programu Mail Manager (program CSQ4TCD1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

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

```

```

    /* failure of the opening of the model */
    /* queue */
    /*-----*/
    MQMErrorHandling( "OPEN TEMPQ", CompCode,
                     Reason );
    ErrorFound = TRUE;
}
return ErrorFound;
}
...

```

Otwieranie istniejącej kolejki

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

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BCA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

    if ((CompCode != MQCC_OK) || (Reason != MQRC_NONE))
    {
        sprintf(pBuff, MESSAGE_4_E,
                ERROR_IN_MQOPEN, CompCode, Reason);
        PrintLine(pBuff);
        RetCode = CSQ4_ERROR;
        goto AbnormalExit1;      /* disconnect processing */
    }

```

```

    }
    :
} /* end of main */

```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQCLOSE w celu zamknięcia kolejki.

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BCA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ. Nazwy i położenia przykładowych aplikacji można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#)  i [Przykładowe programy dla produktu IBM MQ for z/OS](#).

```

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

    /*-----*/
    /* Set up MD structure. */
    /*-----*/
    memset(&MsgDesc, '\0', sizeof(MsgDesc));
    memcpy(MsgDesc.StrucId, MQMD_STRUC_ID,
           sizeof(MsgDesc.StrucId));
    MsgDesc.Version      = MQMD_VERSION_1;
    MsgDesc.Expiry       = MQEI_UNLIMITED;
    MsgDesc.Report       = MQRO_NONE;
    MsgDesc.MsgType      = MQMT_DATAGRAM;
    MsgDesc.Priority     = 1;
}

```

```

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

```

```

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT1

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

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CCB5) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

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

```

```

void Process_Query(void)
{
    /*
    /* Build the reply message
    /*
    :
    /*
    /* Set the object descriptor, message descriptor and

```



```

/* put message options to the values required to      */
/* create the reply message.                          */
/*                                                    */
strncpy(MsgDesc.ObjectName, MsgDesc.ReplyToQ,
        MQ_Q_NAME_LENGTH);
strncpy(MsgDesc.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;
PutBufLen = sizeof(PutBuffer);
MQPUT1(Hconn,
        &ObjDesc,
        &MsgDesc,
        &PutMsgOpts,
        PutBufLen,
        &PutBuffer,
        &CompCode,
        &Reason);

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

```

pobieranie komunikatu

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

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BCA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

/*                                                                    */
GetMsgOpts.Options = MQGMO_NO_WAIT      +                               */
                    MQGMO_BROWSE_FIRST +
                    MQGMO_ACCEPT_TRUNCATED_MSG;

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

```

```

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

```

Pobieranie wiadomości za pomocą opcji wait

W tym przykładzie przedstawiono sposób użycia opcji oczekiwania na wywołanie MQGET.

Ten kod akceptuje obciążone komunikaty. Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CCB5) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

:
MQLONG  Hconn;                /* Connection handle          */
MQHOBJS 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;
MsgBufLen = 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,
      MsgBufLen,
      &MsgBuffer,
      &DataLen,
      &CompCode,
      &Reason);
:
/*
/* Test the output of the MQGET call. If the call
/* failed, send an error message showing the
/* completion code and reason code, unless the
/* reason code is NO_MSG_AVAILABLE.
/*
/*
if (Reason != MQRC_NO_MSG_AVAILABLE)
{
    strncpy(TS_Operation, "MQGET", sizeof(TS_Operation));
    strncpy(TS_ObjName, ObjDesc.ObjectName,
            MQ_Q_NAME_LENGTH);
    Record_Call_Error();
}
:

```

Pobieranie komunikatu przy użyciu sygnalizacji

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

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

```

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

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

    q_ecb = 0;
    GetMsgOpts.Signal1 = &q_ecb;

```

```

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

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

MQGET(Hconn, Hobj, &MsgDesc, &GetMsgOpts,
      BufferLength, message_buffer, &DataLength,
      &CompCode, &Reason);
/*-----*/
/* Check completion and reason codes. */
/*-----*/
switch (CompCode)
{
    case (MQCC_OK):          /* Message retrieved */
        break;
    case (MQCC_WARNING):
        switch (Reason)
        {
            case (MQRC_SIGNAL_REQUEST_ACCEPTED):
                signal_sw = 1;
                break;
            default:
                break; /* Perform error processing */
        }
        break;
    case (MQCC_FAILED):
        switch (Reason)
        {
            case (MQRC_Q_MGR_NOT_AVAILABLE):
            case (MQRC_CONNECTION_BROKEN):
            case (MQRC_Q_MGR_STOPPING):
                break;
            default:
                break; /* Perform error processing. */
        }
        break;
    default:
        break; /* Perform error processing. */
}

/*-----*/
/* If the SET SIGNAL was accepted, set up a loop to */
/* check whether a message has arrived at one second */
/* intervals. The loop ends if a message arrives or */
/* the wait interval specified in the MQGMO */
/* structure has expired. */
/* If a message arrives on the queue, another MQGET */
/* must be issued to retrieve the message. If other */
/* MQM calls have been made in the intervening */
/* period, this may necessitate reinitializing the */
/* MQMD and MQGMO structures. */
/* In this code, no intervening calls */
/* have been made, so the only change required to */
/* the structures is to specify MQGMO_NO_WAIT, */
/* since we now know the message is there. */
/* This code uses the EXEC CICS DELAY command to */
/* suspend the program for a second. A batch program */
/* may achieve the same effect by calling an */
/* assembler language subroutine which issues a */
/* z/OS STIMER macro.
/*-----*/

```

```

if (signal_sw == 1)

```

```

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

```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do zapytania o atrybuty kolejki.

Ten ekstrakt jest pobierane z przykładowej aplikacji Kolejki atrybutów (program CSQ4CCC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

#include <mqc.h>          /* MQ API header file          */
:
#define NUMBEROFSELECTORS 2

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

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

```

```

SelectorsTable[0] = MQIA_INHIBIT_GET;
SelectorsTable[1] = MQIA_INHIBIT_PUT;
/*                                     */
/* Issue the inquire call               */
/* Test the output of the inquire call. If the */
/* call failed, display an error message */
/* showing the completion code and reason code,*/
/* otherwise display the status of the */
/* INHIBIT-GET and INHIBIT-PUT attributes */
/*                                     */
MQINQ(Hconn,
      *pHobj,
      SelectorCount,
      SelectorsTable,
      IntAttrCount,
      IntAttrTable,
      CharAttrLength,
      CharAttr,
      &CompCode,
      &Reason);
if (CompCode != MQCC_OK)
{
    sprintf(Message, MESSAGE_4_E,
            ERROR_IN_MQINQ, CompCode, Reason);
    SetMsg(Message);
}
else
{
    /* Process the changes */
} /* end if CompCode */

```

Ustawianie atrybutów kolejki

W tym przykładzie pokazano, jak można użyć wywołania MQSET w celu zmiany atrybutów kolejki.

Ten ekstrakt jest pobierane z przykładowej aplikacji Kolejki atrybutów (program CSQ4CCC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

const MQHCONN Hconn = MQHC_DEF_HCONN;

static void InhibitGetAndPut(char *Message,
                             PMQHOBJ pHobj,
                             char *Object)
{
    /*                                     */
    /* Declare local variables           */
    /*                                     */
    MQLONG SelectorCount = NUMBEROFSELECTORS;
                                /* Number of selectors */
    MQLONG IntAttrCount = NUMBEROFSELECTORS;
                                /* Number of int attrs */
    MQLONG CharAttrLength = 0;
                                /* Length of char attribute buffer */
    MQCHAR *CharAttr;
                                /* Character attribute buffer */
    MQLONG SelectorsTable[NUMBEROFSELECTORS];
                                /* attribute selectors */
    MQLONG IntAttrTable[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 IntAttrTable to the required status */
    /* - All other variables are already set */
}

```

```

/*                                                    */
SelectorsTable[0] = MQIA_INHIBIT_GET;
SelectorsTable[1] = MQIA_INHIBIT_PUT;
IntAttrsTable[0]  = MQQA_GET_INHIBITED;
IntAttrsTable[1]  = MQQA_PUT_INHIBITED;
:

```

```

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

```

Pobieranie informacji o statusie za pomocą komendy MQSTAT

Ten przykład demonstruje sposób wydawania asynchronicznych operacji MQPUT i pobierania informacji o statusie za pomocą komendy MQSTAT.

Ten ekstrakt jest pobierana z przykładowej aplikacji Calling MQSTAT (program amqsapt0) dostarczane z systemami IBM MQ for Windows . Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

/*****
/*
/* Program name: AMQSAPT0
/*
/* Description: Sample C program that asynchronously puts messages
/*              to a message queue (example using MQPUT & MQSTAT).
/*
/* Licensed Materials - Property of IBM
/*
/* 63H9336
/* (c) Copyright IBM Corp. 2006, 2022. 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                                          */
/*                                                                  */
/*      AMQSAPTO 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 AMQSAPTO start\n");
    if (argc < 2)
    {
        printf("Required parameter missing - queue name\n");
        exit(99);
    }

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

```



```

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

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

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

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

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

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

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

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

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

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

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

```

```

pmo.Options |= MQPMO_ASYNC_RESPONSE;

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

```

Przykłady języka COBOL

Ta kolekcja tematów jest pobierana z przykładowych aplikacji produktu IBM MQ for z/OS . Mają one zastosowanie do wszystkich platform, z wyjątkiem przypadków, gdy jest to zauważone.

Nawiąże połączenie z menedżerem kolejek

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

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BVA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

* -----*

```

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

```

Rozłączanie z menedżerem kolejek

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

Zmienne używane w tym ekstrakcie kodu to te, które zostały ustawione w produkcie [“Nawiąże połączenie z menedżerem kolejek”](#) na stronie 23. Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BVA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu utworzenia kolejki dynamicznej.

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

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

```

```

*      Return to performing section.
*
      EXIT.
      EJECT
*

```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu otwarcia istniejącej kolejki.

Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BVA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQCLOSE.

Zmienne używane w tym ekstrakcie kodu to te, które zostały ustawione w produkcie [“Nawiąże połączenie z menedżerem kolejek”](#) na stronie 23. Ten ekstrakt jest używany z przykładowej aplikacji Przeglądaj (program CSQ4BVA1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT za pomocą kontekstu.

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

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

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



```

:
END-IF.

```

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1.

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB5) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

*
*   API control blocks
*
01 MQM-OBJECT-DESCRIPTOR.
   COPY CMQODV.
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
01 MQM-PUT-MESSAGE-OPTIONS.
   COPY CMQPMOV.
*
* CMQV contains constants (for filling in the
* control blocks) and return codes (for testing
* the result of a call).
*
01 MQM-MQV.
   COPY CMQV SUPPRESS.
* -----*
PROCEDURE DIVISION.
* -----*
:
*   Get the request message.
:
* -----*
PROCESS-QUERY SECTION.
* -----*
:
*   Build the reply message.
:
*
* Set the object descriptor, message descriptor and
* put-message options to the values required to create
* the message.
* Set the length of the message.
*
MOVE MQMD-REPLYTOQ      TO MQOD-OBJECTNAME.
MOVE MQMD-REPLYTOQMGR   TO MQOD-OBJECTQMGRNAME.
MOVE MQMT-REPLY         TO MQMD-MSGTYPE.
MOVE SPACES             TO MQMD-REPLYTOQ.
MOVE SPACES             TO MQMD-REPLYTOQMGR.
MOVE LOW-VALUES         TO MQMD-MSGID.
COMPUTE MQPMO-OPTIONS = MQPMO-SYNCPPOINT +
                      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 MQ0D-OBJECTNAME TO M02-OBJECTNAME
  PERFORM RECORD-CALL-ERROR
  PERFORM FORWARD-MSG-TO-DLQ
END-IF.
*
```

pobieranie komunikatu

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

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

:
* -----*
WORKING-STORAGE SECTION.
* -----*
*
*   W03 - MQM API fields
*
01 W03-HCONN          PIC S9(9) BINARY VALUE ZERO.
01 W03-HOBJ-RESPONSE  PIC S9(9) BINARY.
01 W03-OPTIONS        PIC S9(9) BINARY.
01 W03-BUFFLEN        PIC S9(9) BINARY.
01 W03-DATALEN        PIC S9(9) BINARY.
01 W03-COMPCODE       PIC S9(9) BINARY.
01 W03-REASON         PIC S9(9) BINARY.
*
01 W03-GET-BUFFER.
   05 W03-CSQ4BAM.
   COPY CSQ4VB2.
*
*   API control blocks
*
01 MQM-MESSAGE-DESCRIPTOR.
   COPY CMQMDV.
01 MQM-GET-MESSAGE-OPTIONS.
   COPY 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-SYNCPPOINT +
```

```

MQGMO-ACCEPT-TRUNCATED-MSG +
MQGMO-NO-WAIT.
*
* Set msgid and correlid in MQMD to nulls so that any
* message will qualify.
* Set length to available buffer length.
*
MOVE MQMI-NONE TO MQMD-MSGID.
MOVE MQCI-NONE TO MQMD-CORRELID.
MOVE LENGTH OF W03-GET-BUFFER TO W03-BUFFLEN.
*
CALL 'MQGET' USING W03-HCONN
                  W03-HOBJ-RESPONSE
                  MQMD
                  MQGMO
                  W03-BUFFLEN
                  W03-GET-BUFFER
                  W03-DATALEN
                  W03-COMPCODE
                  W03-REASON.
EVALUATE TRUE
  WHEN W03-COMPCODE NOT = MQCC-FAILED
  :
*   Process the message
  :
  WHEN (W03-COMPCODE = MQCC-FAILED AND
        W03-REASON = MQRC-NO-MSG-AVAILABLE)
    MOVE M01-MESSAGE-9 TO M00-MESSAGE
    PERFORM CLEAR-RESPONSE-SCREEN
*
  WHEN OTHER
    MOVE 'MQGET ' TO M01-MSG4-OPERATION
    MOVE W03-COMPCODE TO M01-MSG4-COMPCODE
    MOVE W03-REASON TO M01-MSG4-REASON
    MOVE M01-MESSAGE-4 TO M00-MESSAGE
    PERFORM CLEAR-RESPONSE-SCREEN
END-EVALUATE.

```

Pobieranie wiadomości za pomocą opcji wait

W tym przykładzie przedstawiono sposób użycia wywołania MQGET z opcją oczekiwania i akceptowanie obciętych komunikatów.

Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB5) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

:
* -----*
WORKING-STORAGE SECTION.
* -----*
*
*   W00 - General work fields
*
01 W00-WAIT-INTERVAL   PIC S9(09) BINARY VALUE 30000.
*
*   W03 - MQM API fields
*
01 W03-HCONN          PIC S9(9) BINARY VALUE ZERO.
01 W03-OPTIONS        PIC S9(9) BINARY.
01 W03-HOBJ-CHECKQ    PIC S9(9) BINARY.
01 W03-COMPCODE       PIC S9(9) BINARY.
01 W03-REASON         PIC S9(9) BINARY.
01 W03-DATALEN        PIC S9(9) BINARY.
01 W03-BUFFLEN        PIC S9(9) BINARY.
*
01 W03-MSG-BUFFER.
05 W03-CSQ4BCAQ.
COPY CSQ4VB3.
*
*   API control blocks
*
01 MQM-MESSAGE-DESCRIPTOR.
COPY CMQMDV.
01 MQM-GET-MESSAGE-OPTIONS.
COPY 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 MQOD-OBJECTNAME TO M02-OBJECTNAME
*       PERFORM RECORD-CALL-ERROR
*   END-IF.
*
*   :

```

Pobieranie komunikatu przy użyciu sygnalizacji

W tym przykładzie przedstawiono sposób korzystania z wywołania MQGET z sygnalizacją. Ten ekstrakt jest pobierana z przykładowej aplikacji Check Check (program CSQ4CVB2) dostarczonej z produktem IBM MQ for z/OS.

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

```

*
* -----*
*   WORKING-STORAGE SECTION.
* -----*
*
*   *   W00 - General work fields
*   :

```

```

01 W00-WAIT-INTERVAL    PIC S9(09) BINARY VALUE 30000.
*
* W03 - MQM API fields
*
01 W03-HCONN            PIC S9(9) BINARY VALUE ZERO.
01 W03-HOBJ-REPLYQ      PIC S9(9) BINARY.
01 W03-COMPCODE         PIC S9(9) BINARY.
01 W03-REASON           PIC S9(9) BINARY.
01 W03-DATALEN          PIC S9(9) BINARY.
01 W03-BUFFLEN          PIC S9(9) BINARY.
:
01 W03-GET-BUFFER.
05 W03-CSQ4BQRM.
COPY CSQ4VB4.
*
05 W03-CSQ4BIIM REDEFINES W03-CSQ4BQRM.
COPY CSQ4VB1.
*
05 W03-CSQ4BPGM REDEFINES W03-CSQ4BIIM.
COPY CSQ4VB5.
:
* API control blocks
*
01 MQM-MESSAGE-DESCRIPTOR.
COPY CMQMDV.
01 MQM-GET-MESSAGE-OPTIONS.
COPY 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-SYNCPPOINT +
                                MQGMO-SET-SIGNAL.
    MOVE W00-WAIT-INTERVAL      TO MQGMO-WAITINTERVAL.
    MOVE LENGTH OF W03-GET-BUFFER TO W03-BUFFLEN.
*
    MOVE ZEROS                  TO L02-REPLY-ECB2.
    SET MQGMO-SIGNAL1 TO ADDRESS OF L02-REPLY-ECB2.

```

```

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

```

```

                                W03-REASON.
*
*   REPLYQ-GETSIGNAL-EXIT.
*
*   Return to performing section.
*
*   EXIT.
*   EJECT
*
*   :

```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do zapytania o atrybuty kolejki.

Ten ekstrakt jest pobierane z przykładowej aplikacji Atrybuty kolejki (program CSQ4CVC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

*
*   Inquire about the attributes.
*
*   CALL 'MQINQ' USING W02-HCONN,
*                       W02-HOBJ,
*                       W02-SELECTORCOUNT,
*                       W02-SELECTORS-TABLE,
*                       W02-INTATTRCOUNT,
*                       W02-INTATTRS-TABLE,
*                       W02-CHARATTRLENGTH,

```

```

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

```

Ustawianie atrybutów kolejki

W tym przykładzie pokazano, jak można użyć wywołania MQSET w celu zmiany atrybutów kolejki.

Ten ekstrakt jest pobierane z przykładowej aplikacji Atrybuty kolejki (program CSQ4CVC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach można znaleźć w sekcji [Przykładowe programy proceduralne \(platformy z wyjątkiem produktu z/OS\)](#).

```

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

```

```

*
* Get the queue name and open the queue.
*
:
*
*
* Initialize the variables required for the set call:
* - Set W02-SELECTORS-TABLE to the attributes to be set
* - Set W02-INTATTRS-TABLE to the required status

```



```

* - All other variables are already set
*
  MOVE MQIA-INHIBIT-GET    TO W02-SELECTORS(1).
  MOVE MQIA-INHIBIT-PUT    TO W02-SELECTORS(2).
  MOVE MQQA-GET-INHIBITED TO W02-INTATTRS(1).
  MOVE MQQA-PUT-INHIBITED TO W02-INTATTRS(2).
*
*   Set the attributes.
*
  CALL 'MQSET' USING W02-HCONN,
                    W02-HOBJ,
                    W02-SELETCOUNT,
                    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 assembler-przykłady języka

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

Nawiąże połączenie z menedżerem kolejek

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

Ten ekstrakt jest pobierana z przykładowego programu przeglądania (CSQ4BAA1) dostarczanego razem z programem IBM MQ for z/OS.

```

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

```

```

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

```

```

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

```

Rozłączanie z menedżerem kolejek

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```

```

BADCALL DS      0H
:
*      CONSTANTS
*
        CMQA
*

```

```

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

```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu utworzenia kolejki dynamicznej.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

:
*
*      R5 = WORK REGISTER.
*
OPEN      DS    0H
*
      MVC  WOD_AREA,MQOD_AREA INITIALIZE WORKING VERSION OF
*                      MQOD WITH DEFAULTS
      MVC  WOD_OBJECTNAME,MOD_Q      COPY IN THE MODEL Q NAME
      MVC  WOD_DYNAMICQNAME,DYN_Q    COPY IN THE DYNAMIC Q NAME
      L     R5,=AL4(MQOO_OUTPUT)     OPEN FOR OUTPUT AND
      A     R5,=AL4(MQOO_INQUIRE)   INQUIRE
      ST    R5,OPTIONS
*
*
*      ISSUE MQI OPEN REQUEST USING REENTRANT
*      FORM OF CALL MACRO
*
      CALL MQOPEN,                X
      (HCONN,                     X
      WOD,                        X
      OPTIONS,                    X
      HOBJ,                       X
      COMPCODE,                   X
      REASON),VL,MF=(E,CALLLIST)
*
      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),VL,MF=L  LIST FORM
                                                OF CALL
*                                                MACRO
:
END

```

Otwieranie istniejącej kolejki

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

Przedstawia on sposób określania dwóch opcji. Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQCLOSE w celu zamknięcia kolejki.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```

```

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

```

```

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT1

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

:
*
*   CONNECT TO QUEUE MANAGER
*
CONN DS 0H
:
*

```

```

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

```

```

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

```

```

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

```

```

:
END

```

pobieranie komunikatu

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

:
*
*   CONNECT TO QUEUE MANAGER
*
CONN    DS   0H
:
*
*   OPEN A QUEUE FOR GET
*
OPEN    DS   0H
:
*
*   R4,R5,R6,R7 = WORK REGISTER.
*
GET     DS   0H
      LA   R4,MQMD                SET UP ADDRESSES AND
      LA   R5,MQMD_LENGTH          LENGTH FOR USE BY MVCL
      LA   R6,WMD                  INSTRUCTION, AS MQMD IS
      LA   R7,WMD_LENGTH           OVER 256 BYES LONG.
      MVCL R6,R4                   INITIALIZE WORKING VERSION
*                                  OF MESSAGE DESCRIPTOR
*
*   MVC  WGM0_AREA,MQGM0_AREA    INITIALIZE WORKING MQGM0
*
      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,CALLLST)
*
      LA   R5,MQCC_OK
      C    R5,COMPCODE
      BNE  BADCALL
*
      :
BADCALL DS   0H
:

```

```

*
*   CONSTANTS
*
      CMQMDA DSECT=NO,LIST=YES
      CMQGM0A 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),VL,MF=L
*
:
:
END

```

Pobieranie wiadomości za pomocą opcji wait

W tym przykładzie przedstawiono sposób użycia opcji oczekiwania na wywołanie MQGET.

Ten kod akceptuje obcięte komunikaty. Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

*
MVC WGMO_AREA,MQGMO_AREA INITIALIZE WORKING MQGMO
L R5,=AL4(MQGMO_WAIT)
A R5,=AL4(MQGMO_ACCEPT_TRUNCATED_MSG)
ST R5,WGMO_OPTIONS
MVC WGMO_WAITINTERVAL,TWO_MINUTES WAIT UP TO TWO
MINUTES BEFORE
FAILING THE
CALL

*
LA R5,BUFFER_LEN RETRIEVE THE BUFFER LENGTH
ST R5,BUFFLEN AND SAVE IT FOR MQM USE

*
* ISSUE MQI GET REQUEST USING REENTRANT FORM OF CALL MACRO
*
* HCONN WAS SET BY PREVIOUS MQCONN REQUEST
* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
CALL MQGET, X
(HCONN, X
HOBJ, X
WMD, X
WGMO, X
BUFFLEN, X
BUFFER, X
DATALEN, X
COMPCODE, X
REASON), X
VL,MF=(E,CALLLST)

*
LA R5,MQCC_OK DID THE MQGET REQUEST
C R5,COMPCODE WORK OK?
BE GETOK YES, SO GO AND PROCESS.
LA R5,MQCC_WARNING NO, SO CHECK FOR A WARNING.
C R5,COMPCODE IS THIS A WARNING?
BE CHECK_W YES, SO CHECK THE REASON.
*

```

```

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

```

```

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

```

```

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

```

Pobieranie komunikatu przy użyciu sygnalizacji

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

:
*
*      CONNECT TO QUEUE MANAGER
*
CONN    DS 0H
        :
*
*      OPEN A QUEUE FOR GET
*
OPEN     DS 0H
        :
*
*      R4,R5,R6,R7 = WORK REGISTER.
*
GET      DS 0H

```

```

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

```

```

*
*   MVC  WGM0_AREA,MQGM0_AREA    INITIALIZE WORKING MQGM0
*   LA   R5,MQGM0_SET_SIGNAL
*   ST   R5,WGM0_OPTIONS
*   MVC  WGM0_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,WGM0_SIGNAL1        AND PUT IT IN THE WORKING
*                               MQGM0
*
*   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
*
WMD       CMQMDA DSECT=NO,LIST=NO
WGMO      CMQGMOA DSECT=NO,LIST=NO
*
CALLLIST  CALL , (0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
:
END

```

Uzyskiwanie informacji o atrybutach kolejki i ustawianie atrybutów kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQINQ w celu sprawdzenia atrybutów kolejki i użycia wywołania MQSET w celu zmiany atrybutów kolejki.

Ten ekstrakt jest pobierane z przykładowej aplikacji Atrybuty kolejki (program CSQ4CAC1) dostarczonej z produktem IBM MQ for z/OS.

```

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

```

```

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

```

```

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

```

```

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

```

```

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

Przykłady PL/I

Korzystanie z języka PL/I jest obsługiwane tylko przez produkt z/OS . Ta kolekcja tematów demonstruje techniki z wykorzystaniem przykładów języka PL/I.

Nawiąże połączenie z menedżerem kolejek

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

Rozłączanie z menedżerem kolejek

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

%INCLUDE SYSLIB(CMQP);
```

```

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

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

```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu utworzenia kolejki dynamicznej.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

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

```

```

        THEN DO;
        :
        CALL ERROR_ROUTINE;
    END;
    ELSE
        DYNAMIC_QUEUE_NAME = LMQOD_OBJECTNAME;

```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQOPEN w celu otwarcia istniejącej kolejki.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

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

```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób korzystania z wywołania MQCLOSE.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```



```

/*****
/* SET CLOSE OPTIONS
*/
/*****
OPTIONS=MQCO_NONE;

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

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT za pomocą kontekstu.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

/*****
/* SET UP LENGTH OF MESSAGE BUFFER AND THE MESSAGE
*/
/*****
BUFLLEN = LENGTH(BUFFER);
BUFFER = PL1_TEST_MESSAGE;
/*****
/*
*/
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.
*/

```

```

/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.          */
/*                                                    */
/*****
CALL MQPUT (HCONN,
            HOBJ,
            LMQMD,
            LMQPMO,
            BUFLLEN,
            BUFFER,
            COMPCODE,
            REASON);

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

```

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

/*****
/* SET UP PUT MESSAGE OPTIONS AS REQUIRED                */
/*****

```

```

LMQPMO.OPTIONS = MQPMO_NO_SYNCPOINT;

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

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

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

```

pobieranie komunikatu

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

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

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

/*****
/*
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.
*/
*****/

```

```

/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.          */
/*                                                    */
/*****
CALL MQGET (HCONN,
            HOBJ,
            LMQMD,
            LMQGMO,
            BUFFERLEN,
            BUFFER,
            DATALEN,
            COMPCODE,
            REASON);

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

```

Pobieranie wiadomości za pomocą opcji wait

W tym przykładzie przedstawiono sposób użycia wywołania MQGET z opcją oczekiwania i akceptowanie obciążonych komunikatów.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

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

/*****
/*

```

```

/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.          */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.            */
/*                                                     */
/*****/
        CALL MQGET (HCONN,
                    HOBJ,
                    LMQMD,
                    LMQGMO,
                    BUFFERLEN,
                    BUFFER,
                    DATALEN,
                    COMPCODE,
                    REASON);

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

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

```

Pobieranie komunikatu przy użyciu sygnalizacji

Ekstrakt kodu demonstrujący sposób użycia wywołania MQGET z sygnalizacją.

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

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

```



```

        CALL NO_MSG;
        :
    END;
    OTHERWISE DO;          /* FAILURE CASE */
/*****
/* ISSUE AN ERROR MESSAGE SHOWING THE COMPLETION CODE */
/* AND THE REASON CODE.          */
*****/
        :
        CALL ERROR_ROUTINE;
        :
    END;

END;

:
END DO_WORK;

GET_MSG: PROC;

```

```

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

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

END GET_MSG;

NO_MSG: PROC;
:
END NO_MSG;

```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do zapytania o atrybuty kolejki.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

```

```

        SELECTORS(01) = MQIA_CURRENT_Q_DEPTH;

        SELECTORCOUNT = 1;
        INTATTRCOUNT = 1;

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

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

```

Ustawianie atrybutów kolejki

W tym przykładzie pokazano, jak można użyć wywołania MQSET w celu zmiany atrybutów kolejki.

Ten ekstrakt nie jest pobierane z przykładowych aplikacji dostarczanych razem z produktem IBM MQ.

```

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

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

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

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

        SELECTORCOUNT = 2;
        INTATTRCOUNT = 2;

        CHARATTRLENGTH = 0;

```



```

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

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

```

State

Informacje uzupełniające zawarte w tej sekcji umożliwiają realizację zadań, które dotyczą potrzeb biznesowych użytkownika.

IBM MQ Pliki COPY, header, include i module

Informacje te stanowią ogólne informacje na temat interfejsu programistycznego.

Ta sekcja zawiera informacje pomocne przy użyciu interfejsu MQI dla różnych języków programowania w następujący sposób.

Pliki nagłówkowe C

Pliki nagłówkowe są udostępniane w celu ułatwienia pisania programów aplikacji C, które korzystają z interfejsu MQI. Te pliki nagłówkowe są podsumowane w tabeli:

Tabela 1. Pliki nagłówkowe C-wywołania prototypów, typów danych, kodów powrotu, stałych i struktur					
Nazwa pliku	Opis	IBM i	Systemy UNIX i Linux®	Windows	z/OS
Wywoływanie prototypów, typów danych, kodów powrotu, stałych i struktur					
CMQC	Definicje MQI	C	C	C	C
CMQBC	Definicje MQAI	C	C	C	
CMQEC	Definicja punktów wejścia interfejsu (w tym CMQC, CMQXC i CMQZC)		C	C	
CMQCFC	Definicje PCF	C	C	C	C
CMQPSC	Definicje publikowania/subskrypcji	C	C	C	C
CMQXC	Definicje kanału i wyjścia	C	C	C	C

Tabela 1. Pliki nagłówkowe C-wywołania prototypów, typów danych, kodów powrotu, stałych i struktur (kontynuacja)

Nazwa pliku	Opis	IBM i	Systemy UNIX i Linux®	Windows	z/OS
CMQZC	Definicje usług instalowalnych	C	C	C	
Klucz: C= udostępnione pliki					

Pliki języka COBOL COPY

Udostępniono różne pliki COPY, które ułatwiają pisanie programów aplikacji w języku COBOL, które korzystają z interfejsu MQI. Te pliki są podsumowane w tabeli:

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury

Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
Kody powrotu i stałe					
CMQx	Definicje MQI	V	V	V	V
CMQCFx	Definicje PCF	V	V	V	V
CMQPSx	Definicje publikowania/subskrypcji	V	V	V	V
CMQXx	Definicje kanału i wyjścia	V	V	V	V
Struktury					
CMQAIRx	MQAIR-rekord informacji uwierzytelniającej		V L	V L	
CMQBOx	MQBO-opcje rozpoczęcia	V L	V L	V L	
CMQCDx	MQCD-definicja kanału	V L	V L	V L	V L
CMQCFBFx	MQCFBF-parametr filtru łańcucha bajtowego PCF	V L	V L	V L	V L
CMQCFBSx	MQCFBS-parametr łańcucha bajtowego PCF	V L	V L	V L	V L
CMQCFGRx	MQCFGR-parametr grupy PCF	V L	V L	V L	V L
CMQCFHx	MQCFH-nagłówek PCF	V L	V L	V L	V L
CMQCFIFx	MQCFIF-parametr filtru liczby całkowitej PCF	V L	V L	V L	V L
CMQCFILx	MQCFIL-parametr listy całkowitej PCF	V L	V L	V L	V L
CMQCFINX	MQCFIN-parametr liczby całkowitej PCF	V L	V L	V L	V L
CMQCFSFx	MQCFSF-parametr filtru łańcucha PCF	V L	V L	V L	V L
CMQCFSLx	MQCFSL-parametr listy łańcuchów PCF	V L	V L	V L	V L
CMQCFSTx	MQCFST-parametr łańcucha PCF	V L	V L	V L	V L

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
CMQCFXLx	MQCFIL64 -64-bitowy parametr listy liczb całkowitych PCF	V L	V L	V L	V L
CMQCFXNx	MQCFIN64 -64-bitowy parametr PCF w postaci liczby całkowitej	V L	V L	V L	V L
CMQCHRVx	MQCHARV-łańcuch o zmiennej długości	V L	V L	V L	V L
CMQCIHx	MQCIH-nagłówek CICS bridge	V L	V L	V L	V L
CMQCNOx	MQCNO-opcje połączenia	V L	V L	V L	V L
CMQCSPx	MQCSP-parametry zabezpieczeń	V L	V L	V L	V L
CMQCXPx	MQCXP-parametry wyjścia kanału	V L			V L
CMQDHx	MQDH-nagłówek dystrybucji	V L	V L	V L	V L
CMQDLHx	MQDLH-nagłówek Dead-letter	V L	V L	V L	V L
CMQDXPx	MQDXP-parametry wyjścia konwersji danych	V L		V L	
CMQEPHx	MQEPH-osadzony nagłówek PCF	V L	V L	V L	V L
CMQGMOx	MQGMO-pobieranie opcji wiadomości	V L	V L	V L	V L
CMQIIHx	MQIIH-nagłówek informacji IMS	V L	V L	V L	V L
CMQMDx	MQMD-deskryptor komunikatu	V L	V L	V L	V L
CMQMD1x	MQMD1 -deskryptor komunikatu w wersji 1	V L	V L	V L	V L
CMQMD2x	MQMD2 -deskryptor komunikatu w wersji 2	V L	V L	V L	V L
CMQMDEx	MQMDE-rozszerzony deskryptor komunikatu	V L	V L	V L	V L
CMQODx	MQOD-deskryptor obiektu	V L	V L	V L	V L
CMQORx	MQOR-rekord obiektu	V L	V L	V L	V L
CMQPMOx	MQPMO-opcje umieszczania komunikatów	V L	V L	V L	V L
CMQRFHx	MQRFH-nagłówek reguł i formatowania	V L	V L	V L	V L
CMQRFH2x	MQRFH2 -reguły i nagłówek formatowania 2	V L	V L	V L	V L
CMQRMHx	MQRMH-nagłówek komunikatu odwołania	V L	V L	V L	V L
CMQRRx	MQRR-rekord odpowiedzi	V L	V L	V L	
CMQSCOx	Opcje konfiguracji MQSCO-SSL		V L	V L	
CMQTMx	MQTM-komunikat wyzwalacza	V L		V L	V L

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
CMQTMx	MQTMC-znak komunikatu wyzwacza	V L	V L		
CMQTM2x	MQTMC2 -komunikat wyzwacza 2 znak	V L	V L	V L	V L
CMQWIx	MQWIH-nagłówek informacji o pracy	V L	V L	V L	V L
CMQXQx	MQXQH-nagłówek kolejki transmisji	V L	V L	V L	V L
Klucz: • Pliki o podanych wartościach początkowych, x = V • Pliki bez podanych wartości początkowych, x = L					

Pliki włączanych PL/I

Następujące pliki INCLUDE są udostępnione dla języka programowania PL/I. Pliki te są dostępne tylko w systemie z/OS .

Tabela 3. PL/I obejmują pliki-typy danych, kody powrotu, stałe i struktury.					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
Typy danych, kody powrotu, stałe i struktury					
CMQP	Definicje MQI				P
CMQCFP	Definicje PCF				P
CMQEPP	Definicje punktów wejścia				P
Komenda CMQPSP	Definicje publikowania/ subskrypcji				P
CMQXP	Definicje kanału i wyjścia				P
Klucz: P= udostępniony plik					

Pliki kopii RPG

Dla języka programowania RPG udostępniono następujące pliki COPY. Te pliki są dostępne tylko w produkcie IBM i.

Tabela 4. Pliki kopii RPG-kody powrotu, stałe i struktury					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
Kody powrotu i stałe					
CMQx	Definicje MQI	G R			
CMQCFx	Definicje PCF	G			
CMQPSx	Definicje publikowania/subskrypcji	G			
CMQXx	Definicje kanału i wyjścia	G R			

Tabela 4. Pliki kopii RPG-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
Struktury					
CMQBOx	MQBO-opcje rozpoczęcia	G H			
CMQCDx	MQCD-definicja kanału	G H R			
CMQCFBFx	MQCFBF-parametr filtra łańcucha bajtowego PCF	G H			
CMQCFBSx	MQCFBS-parametr łańcucha bajtowego PCF	G H			
CMQCFGRx	MQCFGR-parametr grupy PCF	G H			
CMQCFHx	MQCFH-nagłówek PCF	G H			
CMQCFIFx	MQCFIF-parametr filtra liczby całkowitej PCF	G H			
CMQCFILx	MQCFIL-parametr listy całkowitej PCF	G H			
CMQCFINX	MQCFIN-parametr liczby całkowitej PCF	G H			
CMQCFSFx	MQCFSF-parametr filtra łańcucha PCF	G H			
CMQCFSLx	MQCFSL-parametr listy łańcuchów PCF	G H			
CMQCFSTx	MQCFST-parametr łańcucha PCF	G H			
CMQCFXLx	MQCFIL64 -64-bitowy parametr listy liczb całkowitych PCF	G H			
CMQCFXNx	MQCFIN64 -64-bitowy parametr PCF w postaci liczby całkowitej	G H			
CMQCHARVx	MQCHARV-łańcuch o zmiennej długości	G H			
CMQCIHx	MQCIH-nagłówek CICS bridge	G H			
CMQCNOx	MQCNO-opcje połączenia	G H			
CMQCSPx	MQCSP-parametry zabezpieczeń	G H			
CMQCXPx	MQCXP-parametry wyjścia kanału	G H R			
CMQDHx	MQDH-nagłówek dystrybucji	G H R			
CMQDLHx	MQDLH-nagłówek Dead-letter	G H R			
CMQDXPx	MQDXP-parametry wyjścia konwersji danych	G H R			
CMQEPHx	MQEPH-osadzony nagłówek PCF	G H			
CMQGMOx	MQGMO-pobieranie opcji wiadomości	G H R			
CMQIIHx	MQIIH-nagłówek informacji IMS	G H R			

Tabela 4. Pliki kopii RPG-kody powrotu, stałe i struktury (kontynuacja)

Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
CMQMDx	MQMD-deskryptor komunikatu	G H R			
CMQMD1x	MQMD1 -deskryptor komunikatu w wersji 1	G H R			
CMQMD2x	MQMD2 -deskryptor komunikatu w wersji 2	G H			
CMQMDEx	MQMDE-rozszerzony deskryptor komunikatu	G H R			
CMQODx	MQOD-deskryptor obiektu	G H R			
CMQORx	MQOR-rekord obiektu	G H R			
CMQPMOx	MQPMO-opcje umieszczania komunikatów	G H R			
CMQXPx	MQXP-parametry wyjścia routingu publikowania/subskrypcji	G H			
CMQRFHx	MQRFH-nagłówek reguł i formatowania	G H			
CMQRFH2x	MQRFH2 -reguły i nagłówek formatowania 2	G H			
CMQRMHx	MQRMH-nagłówek komunikatu odwołania	G H R			
CMQRRx	MQRR-rekord odpowiedzi	G H R			
CMQTMx	MQTM-komunikat wyzwalacza	G H R			
CMQTMCx	MQTMC-znak komunikatu wyzwalacza	G H R			
CMQTM2x	MQTMC2 -komunikat wyzwalacza 2 znak	G H R			
CMQWIHx	MQWIH-nagłówek informacji o pracy	G H			
CMQXQHx	MQXQH-nagłówek kolejki transmisji	G H R			
Klucz: • Plik dla powiązania statycznego, zainicjowany, udostępniony x = G • Plik dla połączenia statycznego, niezainicjowany, udostępniony x = H • Plik do dynamicznego łączenia, zainicjowany, udostępniony, x = R					

Pliki modułu Visual Basic

Pliki nagłówkowe (lub formularze) są udostępniane w celu ułatwienia pisania programów aplikacji Visual Basic, które korzystają z interfejsu MQI. Te pliki nagłówkowe są dostarczane tylko w 32-bitowych wersjach i są podsumowane w tabeli:

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

z/OS Składanie plików COPY

Udostępniono różne pliki COPY, które ułatwiają pisanie programów aplikacji z/OS Assembler, które korzystają z interfejsu MQI. Te pliki są podsumowane w tabeli:

Tabela 6. z/OS Pliki kopii asemblera-typy danych, kody powrotu, stałe i struktury					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
Typy danych, kody powrotu i stałe					
CMQA	Definicje MQI				A
CMQCFA	Definicje PCF				A
CMQPSA	Definicje publikowania/subskrypcji				A
CMQVERA	Kontrola wersji struktury				A
CMQXA	Definicje kanału i wyjścia				A
Struktury					
CMQCDA	MQCD-definicja kanału				
CMQCFBFA	MQCFBF-parametr filtru łańcucha bajtowego PCF				
CMQCFBSA	MQCFBS-parametr łańcucha bajtowego PCF				A
CMQCFGRA	MQCFGR-parametr grupy PCF				A
CMQCFHA	MQCFH-nagłówek PCF				A
CMQCFIFA	MQCFIF-parametr filtru liczby całkowitej PCF				A
CMQCFILA	MQCFIL-parametr listy całkowitej PCF				A
CMQCFINA	MQCFIN-parametr liczby całkowitej PCF				A
CMQCFSFA	MQCFSF-parametr filtru łańcucha PCF				A
CMQCFSLA	MQCFSL-parametr listy łańcuchów PCF				A

Tabela 6. z/OS Pliki kopii asemblera-typy danych, kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
CMQCFSTA	MQCFST-parametr łańcucha PCF				A
CMQCFXLA	MQCFIL64 -64-bitowy parametr listy liczb całkowitych PCF				A
CMQCFXNA	MQCFIN64 -64-bitowy parametr PCF w postaci liczby całkowitej				A
CMQCHARVA	MQCHARV-łańcuch o zmiennej długości				A
CMQCIHA	MQCIH-nagłówek CICS bridge				A
CMQCNOA	MQCNO-opcje połączenia				A
CMQCSPA	MQCSP-parametry zabezpieczeń				A
CMQCXPA	MQCXP-parametry wyjścia kanału				A
CMQDHA	MQDH-nagłówek dystrybucji				A
CMQDLHA	MQDLH-nagłówek Dead-letter				A
CMQDXPA	MQDXP-parametry wyjścia konwersji danych				A
CMQEPHA	MQEPH-osadzony nagłówek PCF				A
CMQGMOA	MQGMO-pobieranie opcji wiadomości				A
CMQIIHA	MQIIH-nagłówek informacji IMS				A
CMQMDA	MQMD-deskryptor komunikatu				A
CMQMD1A	MQMD1 -deskryptor komunikatu w wersji 1				A
CMQMD2A	MQMD2 -deskryptor komunikatu w wersji 2				A
CMQMDEA	MQMDE-rozszerzony deskryptor komunikatu				A
CMQODA	MQOD-deskryptor obiektu				A
CMQORA	MQOR-rekord obiektu				A
CMQPMOA	MQPMO-opcje umieszczania komunikatów				A
CMQRFHA	MQRFH-nagłówek reguł i formatowania				A
CMQRFH2A	MQRFH2 -reguły i nagłówek formatowania 2				A
CMQRMHA	MQRMH-nagłówek komunikatu odwołania				A
CMQTMA	MQTM-komunikat wyzwalacza				A

Tabela 6. z/OS Pliki kopii asemblera-typy danych, kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	Opis	IBM i	Systemy UNIX	Windows	z/OS
CMQTMCA	MQTMC2 -komunikat wyzwacza 2 znak				A
CMQWCRA	MQWCR-Rekord klastra obciążenia klastra				A
CMQWDRA	MQWDR-rekord miejsca docelowego obciążenia klastra				A
CMQWDR1A	MQWDR1 -rekord miejsca docelowego obciążenia klastra, wersja 1				A
CMQWDR2A	MQWDR2 -rekord miejsca docelowego obciążenia klastra, wersja 2				A
CMQWIHA	MQWIH-nagłówek informacji o pracy				A
CMQWQRA	MQWQR-Rekord kolejki obciążenia klastra				A
CMQWQR1A	MQWQR1 -Rekord kolejki obciążenia klastra, wersja 1				A
CMQWQR2A	MQWQR2 -rekord kolejki obciążenia klastra, wersja 2				A
CMQWXP	MQWXP-parametry wyjścia obciążenia klastra				A
CMQWXP1A	MQWXP1 -parametry wyjścia obciążenia klastra w wersji 1				A
CMQWXP2A	MQWXP2 -parametry wyjścia obciążenia klastra w wersji 2				A
CMQWXP3A	MQWXP3 -parametry wyjścia obciążenia klastra w wersji 3				A
CMQXPA	MQXP-interfejs API CICS -parametry wyjścia krzyżowania				A
CMQXQHA	MQXQH-nagłówek kolejki transmisji				A
CMQXWDA	MQXWD-deskryptor oczekiwania wyjścia				A
Klucz: A= udostępniony plik					

MQ_ * (Długości Łańcucha)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
DŁUGOŚĆ_KODOWANIA_PRODUKTU_MQ	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH,	32	X'00000020'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_APPL_NAME_LENGTH	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'
MQ_APPL_TAG_LENGTH	28	X'0000001C'
MAKSYMALNA_DŁUGOŚĆ MQ_ARM_SUFFIX_LENGTH	2	X'00000002'
DŁUGOŚĆ POPRAWKI MQ_ATTENTION_	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_NAME_LENGTH	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH	8	X'00000008'
MQ_AUTO_REORG_CATALOG_LENGTH	44	X'0000002C'
MQ_AUTO_REORG_TIME_LENGTH	4	X'00000004'
MQ_BATCH_INTERFACE_ID_LENGTH	8	X'00000008'
MQ_DŁUGOŚĆ_NAZWA_HOSTA_Z_MOST	24	X'00000018'
Długość_CODE_kodu_MQ	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH	12	X'0000000C'
MAKSYMALNA_DŁUGOŚĆ_KANAŁU_MQ	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ_KANAŁU MQ_CHANNEL_NAME_LENGTH	20	X'00000014'
MAKSYMALNA_DŁUGOŚĆ_KANAŁU_MQ	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS DŁUGOŚĆ_DŁUGOŚĆ_PLIKU	8	X'00000008'
DŁUGOŚĆ_PRODUKTU_MQ	23	X'00000017'
DŁUGOŚĆ_KLUCZY_MQ	48	X'00000030'
DŁUGOŚĆ_KONTU_MQ	264	X'00000108'
DŁUGOŚĆ_ŁĄCZNA_MQ	128	X'00000080'
DŁUGOŚĆ_POŁĄCZENIE_MQ	24	X'00000018'
DŁUGOŚĆ_MQ_CORREL_LENGTH	24	X'00000018'
DŁUGOŚĆ_DATY_PRODUKTU_MQ	12	X'0000000C'
DŁUGOŚĆ_CZASU_MQ	8	X'00000008'
DŁUGOŚĆ_DANE_MQ	12	X'0000000C'
MQ_DISTINGUISHED_NAME_LENGTH	1024	X'00000400'
DŁUGOŚĆ_GRUPY_MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
MQ_EXIT_DATA_LENGTH	32	X'00000020'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
DŁUGOŚĆ_Z_DŁUGOŚĆ_MQ_NA_DŁUGOŚĆ_MQ	48	X'00000030'
DŁUGOŚĆ_LUB_DŁUGOŚĆ_MQ	(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'
DŁUGOŚĆ_FORMATU_MQ_	8	X'00000008'
MQ_FUNCTION_LENGTH	4	X'00000004'
DŁUGOŚĆ_GRUPY_MQ_GROUP_MQ	24	X'00000018'
DŁUGOŚĆ_HASŁO_LDAP	32	X'00000020'
MQ_LISTENER_NAME_LENGTH	48	X'00000030'
MQ_LISTENER_DESC_LENGTH	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MAKSYMALNA_DŁUGOŚĆ_ŁĄCZENIA_MQ_LTERM_LENGTH	8	X'00000008'
DŁUGOŚĆ_MQ_LU_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ_MQ_LUWID_	16	X'00000010'
MQ_MAX_EXIT_NAME_LENGTH	128	X'00000080'
DŁUGOŚĆ_DŁUGOŚĆ_MCA_MQ	64	X'00000040'
DŁUGOŚĆ_WŁAŚCIWOŚĆ_WŁAŚCIWOŚĆ_MQ	4095	X'00000FFF'
DŁUGOŚĆ_TYPU_MQ	64	X'00000040'
DŁUGOŚĆ_ZADANIA_MCA_MQ	28	X'0000001C'
DŁUGOŚĆ_MAKSYMALNEGO_MQ	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'
Długość_użytkownika_MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	
DŁUGOŚĆ_MAKSYMALNEGO_ŁĄCZENIA_MQ	8	X'00000008'
DŁUGOŚĆ_MODELU_MQ	8	X'00000008'
DŁUGOŚĆ_MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
Długość_ID_MSG_MQ	24	X'00000018'
Długość_TOKEN_MQ	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ_NAZW_NAZW_MQ_NAME	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
DŁUGOŚĆ_OBIEKTU_MQ	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
DŁUGOŚĆ_HASŁA	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'
DŁUGOŚĆ_PROCESU_MQ	48	X'00000030'
DŁUGOŚĆ_DATOWANA_PROCESU_MQ	128	X'00000080'
MQ_PROGRAM_NAME_LENGTH	20	X'00000014'
Wartość parametru MQ_PUT_APPL_NAME_LENGTH	28	X'0000001C'
DŁUGOŚĆ_DATOWANEGO_PRODUKTU_MQ	8	X'00000008'
Długość_czasu MQ_PUT_TIME_LENGTH	8	X'00000008'
MQ_Q_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_DESC_LENGTH	64	X'00000040'
Długość_MQ_Q_MGR_IDENTIFIER_LENGTH	48	X'00000030'
DŁUGOŚĆ_LUB_DŁUGOŚĆ_MQ_Q_MGR_	48	X'00000030'
DŁUGOŚĆ_MQ_Q_NAME_LENGTH	48	X'00000030'
DŁUGOŚĆ_LUB_DŁUGOŚĆ_MQ_QS	4	X'00000004'
DŁUGOŚĆ_ZDALNEJ_PRODUKTU_MQ	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
DŁUGOŚĆ_MQ_SELECTOR_MQ	10240	X'00002800'
MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_COMMAND_LENGTH	255	X'000000FF'
MQ_SERVICE_DESC_LENGTH	64	X'00000040'
MQ_SERVICE_NAME_LENGTH	32	X'00000020'
DŁUGOŚĆ_ŚCIEŻKI_USŁUŻ_MQ_SERVICE_	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH	8	X'00000008'
DŁUGOŚĆ_DŁUGOŚĆ_KONTU_MQ	20	X'00000014'
MQ_SHORT_DNAME_LENGTH	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTOHARDWARE_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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
DŁUGOŚĆ_CZASU	8	X'00000008'
DŁUGOŚĆ_MQ_TOPIC_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ_MQ_TOPIC_NAME_LENGTH	48	X'00000030'
DŁUGOŚĆ_ŁAŃCUCH_MQ	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'
DŁUGOŚĆ_ID_TRANSAKCJI_PRODUKTU_MQ	16	X'00000010'
DŁUGOŚĆ_MQ_TRANSACTION_ID_	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH	64	X'00000040'
MQ_DŁUGOŚĆ_PROGRAMU_TRIGGER_	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH	4	X'00000004'
DŁUGOŚĆ_ID_UŻYTKOWNIKA	12	X'0000000C'
DŁUGOŚĆ_MQ_VERSION_	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

MQ_ * (format komendy-długości łańcucha znaków)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
DŁUGOŚĆ_ARCHIWUM MQ_ARCHIVE_UNIT_LENGTH	8	X'00000008'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_ASID_LENGTH	4	X'00000004'
MQ_AUTH_PROFILE_NAME_LENGTH	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
DŁUGOŚĆ_MQSC_MQ_COMMAND_MQ	32768	X'00008000'
DŁUGOŚĆ_DATOWANA_MQ_DATA_	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
DŁUGOŚĆ_NADAWCZ_MQ	8	X'00000008'
▶ V8.0.0.2 ▶ V8.0.0.2 DŁUGOŚĆ_MQ_ENTITY_NAME_LENGTH	1024	X'00000400'
MQ_ENV_INFO_LENGTH	96	X'00000060'
DŁUGOŚĆ_ADRES_IP	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
MQ_LOG_PATH_LENGTH	1024	X'00000400'
MQ_LRSN_LENGTH	12	X'0000000C'
DŁUGOŚĆ_PRODUKTU_MQ_ORIGIN_LENGTH	8	X'00000008'
DŁUGOŚĆ_Z_NADAWCOWA_	8	X'00000008'
DŁUGOŚĆ_MQ_PST_ID_	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH	4	X'00000004'
DŁUGOŚĆ_ODPOWIEDZI_MQ_RESPONSE_LENGTH	24	X'00000018'
DŁUGOŚĆ_MQ_RBA_	16	X'00000010'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
DŁUGOŚĆ_SYSTEMU_MQ	8	X'00000008'
MAKSYMALNA_DŁUGOŚĆ_ZADANIA_MQ_TASK_	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
DŁUGOŚĆ_ID_PRODUKTU_MQ	256	X'00000100'
DŁUGOŚĆ_UŻYTKOWNIKA_MQ_USER_DATA_	10240	X'00002800'
DŁUGOŚĆ_VOL_MQ	6	X'00000006'

MQACH_* (struktura nagłówka obszaru łańcucha wyjścia funkcji API)

Nazwa	Struktura
MQACH_STRUC_ID,	"ACH"

Nazwa	Struktura
MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQACH_VERSION_1	1	X'00000001'
MQACH_CURRENT_VERSION	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	
MQACH_CURRENT_LENGTH	(value differs by platform or version)	

MQACT_* (Token rozliczania)

Nazwa	Wartość
MQACT_NONE	X'00...00' (32 znaki puste)
MQACT_NONE_ARRAY	'\0','\0',... (32 znaki puste)

MQACT_* (Opcje działania formatu komendy)

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

MQACTP_* (działanie)

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

MQACTT_* (typy znaczników rozliczania)

Nazwa	Wartość szesnastkowa
MQACTT_UNKNOWN	X'00'
MQACTT_CICS_LUOW_ID	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_DEFAULT	X'05'

Nazwa	Wartość szesnastkowa
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_* (Adoptuj nowe typy sprawdzeń MCA i adoptuj nowe typy MCA)

Adoptować nowe sprawdzenia MCA

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQADOPT_CHECK_NONE	0	X'00000000'
MQADOPT_CHECK_ALL	1	X'00000001'
MQADOPT_CHECK_Q_MGR_NAME,	2	X'00000002'
MQADOPT_CHECK_NET_ADDR	4	X'00000004'

Adoptować typy nowego agenta MCA

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQADOPT_TYPE_NO	0	X'00000000'
MQADOPT_TYPE_ALL	1	X'00000001'
MQADOPT_TYPE_SVR	2	X'00000002'
MQADOPT_TYPE_SDR	4	X'00000004'
MQADOPT_TYPE_RCVR	8	X'00000008'
MQADOPT_TYPE_CLUSRCVR	16	X'00000010'

MQAIR_* (struktura rekordu informacji uwierzytelniającej)

Nazwa	Struktura
MQAIR_STRUC_ID	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
MQAIR_CURRENT_VERSION	2	X'00000002'

MQAIT_ * (typ informacji uwierzytelniającej)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAIT_ALL	0	X'00000000'
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'
MQAIT_IDPW_OS	3	X'00000003'
MQAIT_IDPW_LDAP	4	X'00000004'

MQAS_ * (asynchroniczne wartości stanu w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAS_NONE	0	X'00000000'
MQAS_STARTED	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_ZATRZYMANE	3	X'00000003'
MQAS_ZAWIESZONY	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ACTIVE	6	X'00000006'
MQAS_INACTIVE,	7	X'00000007'

MQAT_ * (umieszczania typów aplikacji-Put Application Types)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAT_UNKNOWN	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'
MQAT_OS390	2	X'00000002'
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_WINDOWS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIAN,	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'
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'
UŻYTKOWNIKA_MQAT_	25	X'00000019'
MQAT_BROKER	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
INICJATOR MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'
MQAT_BATCH	32	X'00000020'
MQAT_RRS_BATCH	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	
MQAT_USER_FIRST	65536	X'00010000'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_* (wartości uprawnień dla formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAUTH_NONE	0	X'00000000'
MQAUTH_ALT_USER_AUTHORITY,	1	X'00000001'
MQAUTH_BROWSE	2	X'00000002'
ZMIANA MQAUTH_CHANGE	3	X'00000003'
MQAUTH_CLEAR	4	X'00000004'
MQAUTH_CONNECT,	5	X'00000005'
MQAUTH_CREATE	6	X'00000006'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
Kontekst MQAUTH_PASS_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT,	16	X'00000010'
MQAUTH_CONTROL	17	X'00000011'
MQAUTH_CONTROL_EXTENDED	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUBSCRIBE	20	X'00000014'
MQAUTH_RESUME	21	X'00000015'
MQAUTH_SYSTEM	22	X'00000016'

MQAUTHOPT_* (Opcje uprawnień do formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAUTHOPT_KUMULATYWNE	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'
MQAUTHOPT_ENTITY_SET	2	X'00000002'
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLICIT	16	X'00000010'

MQAXC_* (struktura kontekstu wyjścia funkcji API)

Nazwa	Struktura
MQAXC_STRUC_ID	"AXC¬"
MQAXC_STRUC_ID_ARRAY	'A', 'X', 'C', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAXC_VERSION_1	1	X'00000001'
MQAXC_CURRENT_VERSION	1	X'00000001'

MQAXP_* (struktura parametru wyjścia funkcji API)

Nazwa	Struktura
MQAXP_STRUC_ID	"AXP¬"
MQAXP_STRUC_ID_ARRAY	'A', 'X', 'P', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
MQAXP_CURRENT_VERSION	2	X'00000002'

MQBA_* (selektory atrybutów bajtów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBA_FIRST	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_* (typy parametrów w bajtach w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (długość buforu dla łańcucha mqAddi łańcucha mqSet)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBL_NULL_TERMINATED	-1	X'FFFFFFFF'

MQBMHO_ * (opcje i struktura uchwytu buforu do obsługi komunikatów)

Struktura opcji uchwytu buforu do komunikatu

Nazwa	Struktura
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B','M','H','O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

Opcje Uchwytu Buforu Do Komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIES	1	X'00000001'

MQBND_ * (Powiązania domyślne)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_ * (Opcje początkowe i struktura)

Struktura opcji begin

Nazwa	Struktura
Identyfikator MQBO_STRUC_ID	"BO–"
Tabela MQBO_STRUC_ID_ARRAY	'B','O','–','–'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

Opcje rozpoczęcia

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBO_NONE	0	X'00000000'

MQBT_* (typy mostu w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQBT_OTMA	1	X'00000001'

MQCA_* (selektory atrybutów znakowych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCA_CLUSTER_WORKLOAD_EXIT	2033	X'000007F1'
MQCA_COMMAND_INPUT_Q_NAME	2003	X'000007D3'
MQCA_COMMAND_REPLY_Q_NAME	2067	X'00000813'
MQCA_CREATION_DATE	2004	X'000007D4'
MQCA_CREATION_TIME	2005	X'000007D5'
MQCA_DEAD_LETTER_Q_NAME	2006	X'000007D6'
MQCA_DEF_XMIT_Q_NAME	2025	X'000007E9'
MQCA_DNS_GROUP	2071	X'00000817'
MQCA_ENV_DATA	2007	X'000007D7'
MQCA_FIRST	2001	X'000007D1'
MQCA_IGQ_USER_ID	2041	X'000007F9'
MQCA_INITIATION_Q_NAME	2008	X'000007D8'
MQCA_LAST	4000	X'00000FA0'
MQCA_LAST_USED	2109	X'0000083D'
MQCA_LDAP_PASSWORD	2048	X'00000800'
MQCA_LDAP_USER_NAME	2047	X'000007FF'
MQCA_LU_GROUP_NAME	2072	X'00000818'
MQCA_LU_NAME	2073	X'00000819'
MQCA_LU62_ARM_SUFFIX	2074	X'0000081A'
MQCA_MODEL_DURABLE_Q	2096	X'00000830'
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCA_REMOTE_Q_MGR_NAME	2017	X'000007E1'
MQCA_REMOTE_Q_NAME	2018	X'000007E2'
MQCA_REPOSITORY_NAME	2035	X'000007F3'
MQCA_REPOSITORY_NAMELIST	2036	X'000007F4'
MQCA_RESUME_DATE	2098	X'00000832'
MQCA_RESUME_TIME	2099	X'00000833'
MQCA_SERVICE_DESC	2078	X'0000081E'
MQCA_SERVICE_NAME	2077	X'0000081D'
MQCA_SERVICE_START_ARGS	2080	X'00000820'
MQCA_SERVICE_START_COMMAND	2079	X'0000081F'
MQCA_SERVICE_STOP_ARGS	2082	X'00000822'
MQCA_SERVICE_STOP_COMMAND	2081	X'00000821'
MQCA_STDERR_DESTINATION	2084	X'00000824'
MQCA_STDOUT_DESTINATION	2083	X'00000823'
MQCA_SSL_CRL_NAMELIST	2050	X'00000802'
MQCA_SSL_CRYPT0_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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCA_VERSION	2120	X'00000848'
MQCA_XCF_GROUP_NAME	2043	X'000007FB'
MQCA_XCF_MEMBER_NAME	2044	X'000007FC'
MQCA_XMIT_Q_NAME	2024	X'000007E8'

MQCACF_* (typy parametrów znakowych formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_FIRST	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_FROM_PROCESS_NAME	3003	X'00000BBB'
NAZWA PROCESU 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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
CZAS MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_USER_ID	3039	X'00000BDF'
MQCACF_CHILD_Q_MGR_NAME	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NAME	3043	X'00000BE3'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENT_USER_ID	3045	X'00000BE5'
MQCACF_OBJECT_NAME	3046	X'00000BE6'
MQCACF_EVENT_Q_MGR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NAMES	3048	X'00000BE8'
MQCACF_EVENT_APPL_IDENTITY	3049	X'00000BE9'
MQCACF_EVENT_APPL_NAME	3050	X'00000BEA'
MQCACF_EVENT_APPL_ORIGIN	3051	X'00000BEB'
MQCACF_SUBSCRIPTION_NAME	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY	3054	X'00000BEE'
MQCACF_REG_SUB_IDENTITY,	3055	X'00000BEF'
MQCACF_SUBSCRIPTION_USER_DATA	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_UOW_LOG_START_TIME	3063	X'00000BF7'
MQCACF_UOW_LOG_EXTENT_NAME	3064	X'00000BF8'
MQCACF_PRINCIPAL_ENTITY_NAMES	3065	X'00000BF9'
MQCACF_GROUP_ENTITY_NAMES	3066	X'00000BFA'
MQCACF_AUTH_PROFILE_NAME	3067	X'00000BFB'
MQCACF_ENTITY_NAME	3068	X'00000BFC'
MQCACF_SERVICE_COMPONENT	3069	X'00000BFD'
MQCACF_RESPONSE_Q_MGR_NAME	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_RESTART_LOG_EXTENT_NAME	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
MQCACF_LOG_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN	3079	X'00000C07'
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID,	3081	X'00000C09'
Nazwa MQCACF_PSB_NAME	3082	X'00000C0A'
MQCACF_PST_ID	3083	X'00000C0B'
NUMER_ZADANIA 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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_SYSTEM_NAME	3100	X'00000C1C'
MQCACF_CF_STRUC_BACKUP_START	3101	X'00000C1D'
MQCACF_CF_STRUC_BACKUP_END	3102	X'00000C1E'
MQCACF_CF_STRUC_LOG_Q_MGRS	3103	X'00000C1F'
MQCACF_FROM_STORAGE_CLASS,	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS,	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES	3106	X'00000C22'
MQCACF_DSG_NAME	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
MQCACF_SYSP_CMD_USER_ID	3110	X'00000C26'
MQCACF_SYSP_OTMA_GROUP	3111	X'00000C27'
MQCACF_SYSP_OTMA_MEMBER	3112	X'00000C28'
MQCACF_SYSP_OTMA_DRU_EXIT	3113	X'00000C29'
MQCACF_SYSP_OTMA_TPIPE_PFX	3114	X'00000C2A'
MQCACF_SYSP_ARCHIVE_PFX1	3115	X'00000C2B'
MQCACF_SYSP_ARCHIVE_UNIT1	3116	X'00000C2C'
MQCACF_SYSP_LOG_CORREL_ID	3117	X'00000C2D'
MQCACF_SYSP_UNIT_VOLSER	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_DATE	3120	X'00000C30'
MQCACF_SYSP_Q_MGR_RBA	3121	X'00000C31'
MQCACF_SYSP_LOG_RBA	3122	X'00000C32'
MQCACF_SYSP_SERVICE	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME	3125	X'00000C35'
MQCACF_FROM_SERVICE_NAME	3126	X'00000C36'
Nazwa USŁUGI 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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
CZAS 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'
Identyfikator 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'
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'
FILTR MQCACF	3170	X'00000C62'
MQCACF_BRAK	3171	X'00000C63'
Nazwy MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_LAST_USED	3172	X'00000C64'

MQCACH_* (typy parametrów kanału znaków w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
NAZWA_POŁĄCZENIA_MQCACH_MQ	3506	X'00000DB2'
MQCACH_NAZWA_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'
Nazwa 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'
HASŁO 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'
MQCACH_MCA_NAZWA_ZADANIA	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'
Specyfikacja MQCACH_SSL_CIPHER_SPEC	3544	X'00000DD8'
MQCACH_SSL_PEER_NAME	3545	X'00000DD9'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (deskryptory ADS nagłówka informacji CICS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCADSD_NONE	0	X'00000000'
MQCADSD_SEND,	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_* (wartości powinowactwa połączenia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFEROWANE	1	X'00000001'

MQCAMO_* (typy parametrów monitorowania znaków w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (struktura stałych MQCBC)

Nazwa	Struktura
MQCBC_STRUC_ID	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBF_* (stałe flagi MQCBC)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBF_NONE	0	X'00000000'
MQCBF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_* (stałe MQCBC-typ wywołania zwrotnego)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL,	2	X'00000002'
MQCBCT_REGISTER_CALL,	3	X'00000003'
MQCBCT_DEREGISTER_CALL,	4	X'00000004'
MQCBCT_EVENT_CALL,	5	X'00000005'
MQCBCT_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED	7	X'00000007'

MQCBD_* (struktura stałych MQCBD)

Nazwa	Struktura
MQCBD_STRUC_ID	"CBD¬"

Nazwa	Struktura
MQCBDO_STRUC_ID_ARRAY	'C', 'B', 'D', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBDO_VERSION_1	1	X'00000001'
MQCBDO_CURRENT_VERSION	1	X'00000001'

MQCBDO_* (stałe MQCBDO-opcje wywołania zwrotnego)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBDO_NONE	0	X'00000000'
Wywołania MQCBDO_START_CALL	1	X'00000001'
Wywołanie MQCBDO_STOP_CALL	4	X'00000004'
Wywołanie MQCBDO_REGISTER_CALL	256	X'00000100'
Wywołanie MQCBDO_DEREGISTER_CALL	512	X'00000200'
MQCBDO_FAIL_IF QUIESCING	8192	X'00002000'

MQCBDO_* (Utwórz-Opcje Bag dla mqCreateBag)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBDO_NONE	0	X'00000000'
MQCBDO_USER_BAG	0	X'00000000'
MQCBDO_ADMIN_BAG	1	X'00000001'
MQCBDO_COMMAND_BAG	16	X'00000010'
MQCBDO_SYSTEM_BAG	32	X'00000020'
MQCBDO_GROUP_BAG	64	X'00000040'
MQCBDO_LIST_FORM_ALLOWED	2	X'00000002'
MQCBDO_LIST_FORM_INHIBITED	0	X'00000000'
MQCBDO_REORDER_AS_REQUIRED (mqcb_	4	X'00000004'
MQCBDO_DO_NOT_REORDER,	0	X'00000000'
MQCBDO_CHECK_SELECTORS	8	X'00000008'
MQCBDO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_* (Stałe MQCBT Jest to typ funkcji Callback)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_* (kody zakończenia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCC_OK	0	X'00000000'
MQCC_WARNING,	1	X'00000001'
MQCC_FAILED	2	X'00000002'
MQCC_UNKNOWN	-1	X'FFFFFFFF'

MQCCSI_* (Identyfikatory Kodowanego Zestawu Znaków)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_DEFAULT	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'
MQCCSI_APPL	-3	X'FFFFFFFD'

MQCCT_* (CICS -Opcje zadania konwersacyjnego w nagłówku informacji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_* (struktura definicji kanału)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
MQCD_VERSION_11	11	X'0000000B'
MQCD_CURRENT_VERSION	11	X'0000000B'
MQCD_LENGTH_4	(value differs by platform or version)	

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_LENGTH_10	(value differs by platform or version)	
MQCD_LENGTH_11	(value differs by platform or version)	
MQCD_CURRENT_LENGTH	(value differs by platform or version)	

MQCDC_ * (konwersja danych kanału)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION	0	X'00000000'

MQCERT_ * (typ strategii sprawdzania poprawności certyfikatu)

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

MQCF_ * (Flagi Możliwości)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_* (narzędzie nagłówka informacji CICS)

Nazwa	Wartość szesnastkowa
MQCFAC_NONE	X'00...00' (8 zer)
MQCFAC_NONE_ARRAY	'\0', '\0', ... (8 zer)

MQCFBF_* (struktura parametru filtru łańcucha bajtów w formacie wiersza komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_* (struktura parametru łańcucha bajtowego formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFCC_* (opcje sterujące nagłówka formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFCC_LAST	1	X'00000001'
MQCFCC_NOT_LAST	0	X'00000000'

MQCFGR_* (struktura parametru grupy formatów komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_* (struktura nagłówka formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFH_STRUC_LENGTH	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
MQCFH_CURRENT_VERSION	3	X'00000003'

MQCFIF_* (struktura parametru filtru liczby całkowitej w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIF_STRUC_LENGTH	20	X'00000014'

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

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (struktura parametru listy liczb całkowitych w formacie 64-bitową komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_* (struktura parametru w postaci liczby całkowitej w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIN_STRUC_LENGTH	16	X'00000010'

MQCFIN64_* (format komendy 64-bitowej-struktura parametru liczby całkowitej)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIN64_STRUC_LENGTH	24	X'00000018'

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

Opcje odświeżania repozytorium formatu komend

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

Opcje usuwania kolejek w formacie komendy

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

MQCFOP_* (Operatory filtru formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_GREATER	4	X'00000004'
MQCFOP_NOT_LESS	6	X'00000006'
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NOT_GREATER	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE	21	X'00000015'
MQCFOP_CONTAINS	10	X'0000000A'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFOP_EXCLUDES	13	X'0000000D'
MQCFOP_CONTAINS_GEN	26	X'0000001A'
MQCFOP_EXCLUDES_GEN	29	X'0000001D'

MQCFR_* (odzyskiwalność systemu CF)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFR_TAK	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_* (struktura parametru filtru łańcucha formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_* (struktura parametru listy łańcuchów formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_* (struktura parametru łańcucha formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_* (Status komendy CF-status)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFSTATUS_NOT_FOUND	0	X'00000000'
MQCFSTATUS_ACTIVE	1	X'00000001'
MQCFSTATUS_IN_RECOVER	2	X'00000002'
MQCFSTATUS_IN_BACKUP	3	X'00000003'
Funkcja MQCFSTATUS_FAILED	4	X'00000004'
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_UNKNOWN	6	X'00000006'
MQCFSTATUS_ADMIN_NIEPEŁNY	20	X'00000014'
MQCFSTATUS_NEVER_USED,	21	X'00000015'
MQCFSTATUS_NO_BACKUP	22	X'00000016'
MQCFSTATUS_NOT_FAILED	23	X'00000017'
MQCFSTATUS_NOT_RECOVERABLE	24	X'00000018'
MQCFSTATUS_XES_ERROR (BŁĄD)	25	X'00000019'

MQCFT_* (typy formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
RAPORT 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'
Statystyki MQCFT_STATISTICS	21	X'00000015'
MQCFT_ACCOUNTING	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_* (typy CF w formacie komendy)

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

MQCFUNC_* (funkcje nagłówka informacji CICS)

Nazwa	Struktura
MQCFUNC_MQCONN	"CONN"
MQCFUNC_MQGET-MQGET	"GET~"
MQCFUNC_MQINQ	"INQ~"

Nazwa	Struktura
MQCFUNC_MQOPEN	"OPEN"
MQCFUNC_MQPUT 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	'-','-','-','-'

Uwaga: Symbol - reprezentuje pojedynczy pusty znak.

MQCGWI_* (CICS przedział czasu oczekiwania na pobranie nagłówka informacji)

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

MQCHAD_* (automatyczna definicja kanału)

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

MQCHIDS_* (Status niepewny formatu komendy)

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

MQCHLD_* (Sdyspozycje kanału formatu komendy)

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

MQCHS_ * (Status kanału formatu komendy-Command format Channel Status)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHS_INACTIVE	0	X'00000000'
MQCHS_BINDING	1	X'00000001'
MQCHS_STARTING	2	X'00000002'
MQCHS_RUNNING	3	X'00000003'
MQCHS_STOPPING	4	X'00000004'
MQCHS_RETRYING	5	X'00000005'
MQCHS_STOPPED	6	X'00000006'
MQCHS_REQUESTING	7	X'00000007'
MQCHS_PAUSED	8	X'00000008'
MQCHS_INITIALIZING	13	X'0000000D'
MQCHS_SWITCHING	14	X'0000000E'

MQCHSH_ * (opcje współużytkowanego restartowania kanału formatu komend)

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

MQCHSR_ * (Opcje zatrzymania kanału w formacie komendy)

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

MQCHSSTATE_ * (Podstany kanału kanału komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHSSTATE_OTHER	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_WYSYŁANIE	200	X'000000C8'
MQCHSSTATE_OTRZYMUJĄCYCH	300	X'0000012C'
MQCHSSTATE_SERIALIZOWANIE	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'
MQCHSSTATE_HEARTBIĆ	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT	1100	X'0000044C'
MQCHSSTATE_IN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONNECTING	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT,	1500	X'000005DC'
MQCHSSTATE_IN_MQGET,	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRESSING	1800	X'00000708'

MQCHT_* (typy kanałów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHT_SENDER	1	X'00000001'
SERWER_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_* (typy tabel kanału formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_* (identyfikator korelacji)

Nazwa	Wartość
MQCI_NONE	X'00...00' (24 znaki puste)
MQCI_NONE_ARRAY	'\0','\0',... (24 znaki puste)
MQCI_NOWA_SESJA	X'414D5121...'
MQCI_NEW_SESSION_ARRAY	'\x41','\x4D','\51','\x21',...

MQCIH_* (struktura nagłówka informacyjnego produktu CICS i flagi)

Struktura nagłówka informacyjnego produktu CICS

Nazwa	Struktura
MQCIH_STRUC_ID	"CIH-"
MQCIH_STRUC_ID_ARRAY	'C','I','H','-'

Uwaga: Symbol - reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_CURRENT_VERSION	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

Flagi nagłówka informacji CICS

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

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

MQCLRS_* (format komendy Wyczyść zasięg łańcucha tematu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_ * (format komendy Wyczyść typ łańcucha tematu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCLRT_ZACHOWANE	1	X'00000001'

MQCLT_ * (typy odsyłaczy nagłówek informacyjnego produktu CICS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
PROGRAM MQCLT_PROGRAM	1	X'00000001'
MQCLT_TRANSACTION,	2	X'00000002'

MQCLWL_ * (obciążenie klastra)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFFD'

MQCLXQ_ * (Typ kolejki transmisji klastra)

MQCLXQ_* to wartości, które można ustawić w atrybucie menedżera kolejek DEFCLXQ . Atrybut DEFCLXQ określa, która kolejka transmisji jest wybierana domyślnie przez kanały wysyłające klastry w celu pobierania komunikatów z kanałów odbiorczych klastra.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Odsyłacze pokrewne

[“DefClusterXmitQueueTyp \(MQLONG\)” na stronie 826](#)

Atrybut DefClusterXmitQueueTyp określa, która kolejka transmisji jest wybierana domyślnie przez kanały wysyłające klastry w celu pobierania komunikatów z, do kanałów odbierających klastry.

[“MQINQ-zapytanie o atrybuty obiektu” na stronie 714](#)

Wywołanie funkcji MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Informacje pokrewne

[Zmiana menedżera kolejek](#)

[Zapytaj menedżera kolejek](#)

[Sprawdzanie menedżera kolejek \(odpowiedź\)](#)

MQCMD_ * (kody komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_BRAK	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
Proces MQCMD_CHANGE_PROCESS	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_CREATE_PROCESS	5	X'00000005'
MQCMD_DELETE_PROCESS	6	X'00000006'
Proces MQCMD_INQUIRE_PROCESS	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
Kolejka 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'
Kanał MQCMD_COPY_CHANNEL	22	X'00000016'
MQCMD_CREATE_CHANNEL	23	X'00000017'
MQCMD_DELETE_CHANNEL	24	X'00000018'
MQCMD_INQUIRE_CHANNEL	25	X'00000019'
Kanał MQCMD_PING_CHANNEL	26	X'0000001A'
MQCMD_RESET_CHANNEL	27	X'0000001B'
Kanał MQCMD_START_CHANNEL	28	X'0000001C'
Kanał MQCMD_STOP_CHANNEL	29	X'0000001D'
MQCMD_START_CHANNEL_INIT	30	X'0000001E'
MQCMD_START_CHANNEL_LISTENER	31	X'0000001F'
Lista nazw MQCMD_CHANGE_NAMELIST	32	X'00000020'
MQCMD_COPY_NAMELIST	33	X'00000021'
MQCMD_CREATE_NAMELIST,	34	X'00000022'
MQCMD_DELETE_NAMELIST	35	X'00000023'
Tabela MQCMD_INQUIRE_NAMELIST	36	X'00000024'
MQCMD_INQUIRE_NAMELIST_NAMES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVE_CHANNEL	39	X'00000027'
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_INQUIRE_Q_STATUS	41	X'00000029'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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, PUBLIKATOR	61	X'0000003D'
Subskrybent komendy MQCMD_DEREGISTER_SUBSKRYBENTA	62	X'0000003E'
MQCMD_PUBLISH	63	X'0000003F'
MQCMD_REGISTER_PUBLISHER	64	X'00000040'
Subskrybent MQCMD_REGISTER_SUBSKRYBENTA	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'
KLASTER_MENEDŻERA_KOLEJEK MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
Klaster_menedżera_kolejek MQCMD_SUSPEND_Q_MGR_CLUSTER	72	X'00000048'
Klaster MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_KLASTRA	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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_LOGGER_EVENT,	91	X'0000005B'
MQCMD_RESET_Q_MGR	92	X'0000005C'
MQCMD_CHANGE_LISTENER	93	X'0000005D'
OBIEKT NASŁUCHIWANIA MQCMD_COPY_LISTENER	94	X'0000005E'
MQCMD_CREATE_LISTENER	95	X'0000005F'
MQCMD_DELETE_LISTENER	96	X'00000060'
MQCMD_INQUIRE_LISTENER	97	X'00000061'
MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_INQUIRE_USAGE	126	X'0000007E'
Kolejka MQCMD_MOVE_Q	127	X'0000007F'
MQCMD_RECOVER_BSDFS	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'
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'
STATUS_MENEDŻERA_KOLEJEK MQCMD_INQUIRE_Q_MGR_STATUS	161	X'000000A1'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_CREATE_LOG	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
Kanał 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'
Temat MQCMD_CHANGE_TOPIC	170	X'000000AA'
Temat MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREATE_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC	173	X'000000AD'
MQCMD_INQUIRE_TOPIC	174	X'000000AE'
MQCMD_INQUIRE_TOPIC_NAMES,	175	X'000000AF'
Subskrypcja MQCMD_INQUIRE_SUBSCRIPTION	176	X'000000B0'
SUBSKRYPCJA_MQCMD_CREATE_SUBSKRYPCJI	177	X'000000B1'
SUBSKRYPCJA mqcmd_change_subskrypcji	178	X'000000B2'
MQCMD_DELETE_SUBSCRIPTION	179	X'000000B3'
Subskrypcja MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_INQUIRE_SUB_STATUS	182	X'000000B6'
MQCMD_INQUIRE_TOPIC_STATUS,	183	X'000000B7'
Łańcuch MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_INQUIRE_PUBSUB_STATUS	185	X'000000B9'
Kanał MQCMD_PURGE_CHANNEL	195	X'000000C3'

MQCMDI_ * (wartości informacji o komendzie w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
ZAAKCEPTOWANO wartość MQCMDI_CMDSCOPE_ACCEPTED	1	X'00000001'
WYGENEROWANO mqcmdi_cmdscope_generated	2	X'00000002'
Komenda MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
MQCMDI_COMMAND_ACCEPTED	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_W_KOLEJCE	6	X'00000006'
MQCMDI_CHANNEL_INIT_STARTED	7	X'00000007'
Komenda MQCMDI_RECOVER_STARTED	11	X'0000000B'
MQCMDI_BACKUP_STARTED	12	X'0000000C'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMDI_RECOVER_COMPLETED	13	X'0000000D'
MQCMDI_SEC_TIMER_ZERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
BŁĄD 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_WIELKIMI (wielkie litery)	21	X'00000015'
MQCMDI_SEC_MIXEDCASE	22	X'00000016'

MQCMDL_* (poziomy komend)

Nazwa	Wartość
MQCMDL_LEVEL_600	600
MQCMDL_LEVEL_700	700
MQCMDL_LEVEL_701	701
MQCMDL_LEVEL_710	710
MQCMDL_LEVEL_711	711
MQCMDL_LEVEL_750	750
MQCMDL_LEVEL_800	800
  MQCMDL_LEVEL_801	801
  MQCMDL_LEVEL_802	802

Uwagi:

1. Produkt MQCMDL_LEVEL_801 ma zastosowanie tylko do produktów Platformy UNIX i IBM ipodczas instalowania produktu IBM MQ 8.0.0, Fix Pack 2.
2.  Program MQCMDL_LEVEL_802 ma zastosowanie tylko do produktów Windows, Platformy UNIX i IBM ipodczas instalowania produktu IBM MQ Wersja 8.0.0, pakiet poprawek 3

MQCMHO_* (Utwórz opcje i strukturę uchwytu komunikatu)

Utwórz strukturę opcji uchwytu komunikatu

Nazwa	Struktura
MQCMHO_STRUC_ID,	"CMHO"
Tablica MQCMHO_STRUC_ID_ARRAY	'C', 'M', 'H', 'O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMHO_VERSION_1	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMHO_CURRENT_VERSION	1	X'00000001'

Opcje tworzenia uchwytu komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'
MQCMHO_VALIDATE	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCN*_*(opcje połączenia i struktura)

Struktura opcji łączenia

Nazwa	Struktura
MQCN*_STRUC_ID	"CNO~"
MQCN*_STRUC_ID_ARRAY	'C','N','O','~'

Uwaga: Symbol ~ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCN*_VERSION_1	1	X'00000001'
MQCN*_VERSION_2	2	X'00000002'
MQCN*_VERSION_3	3	X'00000003'
MQCN*_VERSION_4	4	X'00000004'
MQCN*_VERSION_5	5	X'00000005'
MQCN*_CURRENT_VERSION	5	X'00000005'

Opcje połączenia

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCN*_STANDARD_BINDING	0	X'00000000'
MQCN*_FASTPATH_BINDING	1	X'00000001'
MQCN*_SERIALIZE_CONN_TAG_Q_MGR	2	X'00000002'
MQCN*_SERIALIZE_CONN_TAG_QSG	4	X'00000004'
MQCN*_RESTRICT_CONN_TAG_Q_MGR	8	X'00000008'
MQCN*_RESTRICT_CONN_TAG_QSG	16	X'00000010'
MQCN*_HANDLE_SHARE_NONE	32	X'00000020'
MQCN*_HANDLE_SHARE_BLOCK	64	X'00000040'
MQCN*_HANDLE_SHARE_NO_BLOCK	128	X'00000080'
MQCN*_SHARED_BINDING	256	X'00000100'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCNO_ISOLATED_BINDING	512	X'00000200'
MQCNO_LOCAL_BINDING	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_OUTPUT_ONLY	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT,	16777216	X'01000000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_* (opcje zamknięcia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (długość danych wyjściowych nagłówka informacji CICS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_* (kompresja kanału)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCOMPRESS_NOT_AVAILABLE	-1	X'FFFFFFFF'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_* (identyfikator połączenia)

Nazwa	Wartość
MQCONNID_BRAK	X'00...00' (24 znaki puste)
MQCONNID_NONE_ARRAY	'\0', '\0', ... (24 znaki puste)

MQCOPY_* (Opcje kopiowania właściwości)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCOPY_BRAK	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
ODPOWIEDŹ MQCOPY_REPLY	8	X'00000008'
RAPORT MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_* (typy kolejek klastrów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTE_Q	3	X'00000003'
Alias_menedżera_kolejek MQCQT_Q_MGR_ALIAS	4	X'00000004'

MQCRC_* (kody powrotu nagłówka informacyjnego produktu CICS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCRC_OK	0	X'00000000'
MQCRC_CICS_EXEC_ERROR	1	X'00000001'
BŁĄD MQCRC_MQ_API_ERROR	2	X'00000002'
MQCRC_BRIDGE_ERROR	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (stałe MQCBC-stan konsumenta)

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

MQCSC_* (CICS Kody początkowe nagłówka informacji)

Nazwa	Struktura
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	'-', '-', '-', '-', '-'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

MQCSP_* (Struktura parametrów zabezpieczeń połączenia i typy uwierzytelniania)

Struktura parametrów zabezpieczeń połączenia

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

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSP_VERSION_1	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSP_CURRENT_VERSION	1	X'00000001'

Parametry zabezpieczeń połączenia-typy uwierzytelniania

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'

MQCSRV_* (opcje serwera komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSRV_CONVERT_NO	0	X'00000000'
MQCSRV_CONVERT_YES	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_YES	1	X'00000001'

MQCT_* (znacznik połączenia menedżera kolejek)

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

MQCTES_* (Status zakończenia zadania nagłówek informacyjnego produktu CICS)

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

MQCTLO_* (struktura opcji MQCTL i opcje kontroli konsumenta)

Struktura opcji MQCTL

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

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

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

Opcje MQCTL opcji kontroli konsumenta

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

MQCUOWC_* (elementy sterujące jednostki nagłówka informacji o produkcji CICS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCUOWC_ONLY	273	X'00000111'
MQCUOWC_CONTINUE	65536	X'00010000'
MQCUOWC_FIRST	17	X'00000011'
MQCUOWC_MIDDLE	16	X'00000010'
MQCUOWC_LAST	272	X'00000110'
MQCUOWC_COMMIT	256	X'00000100'
MQCUOWC_BACKOUT	4352	X'00001100'

MQCXP_* (struktura parametru wyjścia kanału)

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

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

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

MQDC_* (klasa docelowa)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDC_MANAGED	1	X'00000001'
Zmaterializowana MQDC_XX_ENCODE_CASE_ONE udostępniona	2	X'00000002'

MQDCC_* (opcje konwersji, maski i czynniki)

Opcje konwersji

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

Maski opcji konwersji i czynniki

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

MQDELO_* (opcje usuwania publikowania/subskrypcji)

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

MQDH_* (Struktura nagłówka dystrybucji)

Nazwa	Struktura
MQDH_STRUC_ID,	"DH¬¬"
Tablica MQDH_STRUC_ID_ARRAY	'D', 'H', '¬', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_* (flagi nagłówka dystrybucji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_* (typy rozłączeń w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDISCONNECT_NORMAL,	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_* (listy dystrybucyjne)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_* (struktura nagłówka Dead-letter)

Nazwa	Struktura
MQDLH_STRUC_ID	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D', 'L', 'H', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDLH_VERSION_1	1	X'00000001'
MQDLH_CURRENT_VERSION	1	X'00000001'

MQDLV_* (dostarczanie komunikatów trwa/nietrwałych)

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

MQDMHO_* (Usuń opcje i strukturę uchwytu komunikatu)

Usuń strukturę opcji uchwytu komunikatu

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

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

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

Opcje usuwania uchwytu komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMHO_NONE	0	X'00000000'

MQDMPO_* (Usuń opcje i strukturę właściwości komunikatu)

Usuń strukturę opcji właściwości komunikatu

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

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

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

Opcje usuwania właściwości komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_* (WLM-WLM)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_YES	1	X'00000001'

MQDT_* (typy docelowe)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_* (struktura parametru wyjścia konwersji)

Nazwa	Struktura
MQDXP_STRUC_ID	"DXP¬"
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_* (wartości Signal)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEC_MSG_PRZYBYŁ	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED	3	X'00000003'
MQEC_WAIT_ANULOWANE	4	X'00000004'
MQEC_Q_MGR QUIESCING,	5	X'00000005'
MQEC_CONNECTION QUIESCING	6	X'00000006'

MQEI_* (utrata ważności)

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

MQENC_* (kodowanie)

MQENC_* (kodowanie)

Nazwa	Platforma	Wartość dziesiętna	Wartość szesnastkowa
MQENC_NATIVE	IBM i	273	X'00000111 '
	Linux	546	X'00000222 '
	Linux na platformie SPARC	273	X'00000111 '
	Linux na platformie x86	546	X'00000222 '
	Solaris na platformie SPARC	273	X'00000111 '
	Systemy UNIX	273	X'00000111 '
	Windows	546	X'00000222 '
	Micro Focus COBOL w systemie Windows	17	X'00000011 '
	z/OS	785	X'00000311 '

MQENC_* (maski kodowania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_INTEGER_MASK	15	X'0000000F '
MQENC_DECIMAL_MASK	240	X'000000F0 '
MQENC_FLOAT_MASK	3840	X'00000F00 '
Maska MQENC_RESERVED_MASK	-4096	X'FFFFFF000 '

MQENC_* (Encodings for Binary Integers)

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

MQENC_* (Encodings for Packed Decimal Integer)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_DECIMAL_UNDEFINED	0	X'00000000 '
MQENC_DECIMAL_NORMAL	16	X'00000010 '
MQENC_DECIMAL_REVERSED	32	X'00000020 '

MQENC_ * (kodowania dla liczb zmiennopozycyjne)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQENC_FLOAT_TNS	1024	X'00000400'

MQEPH_ * (wbudowana struktura nagłówka formatu komend i flagi)

Struktura nagłówka osadzonego formatu komend

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

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

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

Flagi nagłówka osadzonego formatu komend

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED,	1	X'00000001'

MQET_ * (typy Escape-format komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQET_MQSC	1	X'00000001'

MQEVO_ * (pochodzenie zdarzenia w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEVO_INNE	0	X'00000000'
KONSOLA MQEVO_CONSOLE	1	X'00000001'
WYWOŁANIE mqevo_init	2	X'00000002'
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'

MQEVR_* (rejestrwanie zdarzeń w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPTION	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_* (przedział czasu skanowania dat ważności)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEXPI_OFF	0	X'00000000'

MQFB_* (wartości opinii)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
BŁĄD 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'
Komunikat MQFB_NOT_A_REPOSITORY_MSG	280	X'00000118'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
DZIAŁANIA MQFB_MAX_ACTIVITIES	282	X'0000011A'
MQFB_NOT_FORWARDED	283	X'0000011B'
MQFB_NOT_DELIVERED	284	X'0000011C'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQFB_UNSUPPORTED_FORWARDING	285	X'0000011D'
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
MQFB_DATA_LENGTH_ZERO	291	X'00000123'
MQFB_DŁUGOŚĆ_DŁUGOŚĆ_UJEMNEGO	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'
BŁĄD MQFB_IH_ERROR	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
BŁĄD MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_INTERNAL_ERROR	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'
MQFB_CICS_NIEPOWODZENIE bridge_failure	403	X'00000193'
MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'
MQFB_CICS-BŁĄD ID_CCSID	405	X'00000195'
MQFB_CICS_BŁĄD __ENCODING_ERROR	406	X'00000196'
MQFB_CICS-BŁĄD CIH_ERROR	407	X'00000197'
MQFB_CICS_UOW_ERROR	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
ŻĄDANIE MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIBER_IS_PUBLISHER	502	X'000001F6'
Niezgodność MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
Niezgodność MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'
MQFB_SYSTEM_LAST	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_* (opcje wymuszenia formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQFC_YES	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_* (Formaty)

Nazwa	Wartość
MQFMT_NONE	"~~~~~"
ADMINISTRATOR MQFMT_ADMIN	"MQADMIN~"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM~"
MQFMT_CICS	"MQCICS~~"
MQFMT_COMMAND_1	"MQCMD1~~"
MQFMT_COMMAND_2	"MQCMD2~~"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD~~"
MQFMT_DIST_HEADER	"MQHDIST~"
MQFMT_EMBEDDED_PCF	"MQHEPCF~"
Zdarzenie MQFMT_EVENT	"MQEVENT~"
MQFMT_IMS	"MQIMS~~~"
MQFMT_IMS_VAR_STRING	"MQIMSVS~"
MQFMT_MD_EXTENSION	"MQHMDE~~"
MQFMT_PCF	"MQPCF~~~"
MQFMT_REF_MSG_HEADER	"MQHREF~~"
MQFMT_RF_HEADER	"MQHRF~~~"
MQFMT_RF_HEADER_1	"MQHRF~~~"
MQFMT_RF_HEADER_2	"MQHRF2~~"
MQFMT_STRING	"MQSTR~~~"
MQFMT_TRIGGER	"MQTRIG~~"
Nagłówek MQFMT_WORK_INFO_HEADER	"MQHWIH~~"
MQFMT_XMIT_Q_HEADER	"MQXMIT~~"
MQFMT_NONE_ARRAY	'~','~','~','~','~','~','~','~','~'
MQFMT_ADMIN_ARRAY	'M','Q','A','D','M','I','N','~'
MQFMT_CHANNEL_COMPLETED_ARRAY	'M','Q','C','H','C','O','M','~'
MQFMT_CICS_ARRAY	'M','Q','C','I','C','S','~','~'
MQFMT_COMMAND_1_ARRAY	'M','Q','C','M','D','1','~','~'
MQFMT_COMMAND_2_ARRAY	'M','Q','C','M','D','2','~','~'
MQFMT_DEAD_LETTER_HEADER_ARRAY	'M','Q','D','E','A','D','~','~'

Nazwa	Wartość
MQFMT_DIST_HEADER_ARRAY	'M','Q','H','D','I','S','T','¬'
MQFMT_EMBEDDED_PCF_ARRAY	'M','Q','H','E','P','C','F','¬'
MQFMT_EVENT_ARRAY	'M','Q','E','V','E','N','T','¬'
MQFMT_IMS_ARRAY	'M','Q','I','M','S','¬','¬','¬'
MQFMT_IMS_VAR_STRING_ARRAY	'M','Q','I','M','S','V','S','¬'
MQFMT_MD_EXTENSION_ARRAY	'M','Q','H','M','D','E','¬','¬'
MQFMT_PCF_ARRAY	'M','Q','P','C','F','¬','¬','¬'
MQFMT_REF_MSG_HEADER_ARRAY	'M','Q','H','R','E','F','¬','¬'
MQFMT_RF_HEADER_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_1_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_2_ARRAY	'M','Q','H','R','F','2','¬','¬'
MQFMT_STRING_ARRAY	'M','Q','S','T','R','¬','¬','¬'
MQFMT_TRIGGER_ARRAY	'M','Q','T','R','I','G','¬','¬'
MQFMT_WORK_INFO_HEADER_ARRAY	'M','Q','H','W','I','H','¬','¬'
MQFMT_XMIT_Q_HEADER_ARRAY	'M','Q','X','M','I','T','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

MQGA_* (selektory atrybutów grupy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQGA_FIRST	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_* (typy parametrów grupy formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
DANE MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_USED	8012	X'00001F4C'

MQGI_* (identyfikator grupy)

Nazwa	Wartość
MQGI_NONE	X'00...00' (24 znaki puste)
MQGI_NONE_ARRAY	'\0','\0',... (24 znaki puste)

MQGMO_* (Pobranie opcji i struktury komunikatu)

Pobierz strukturę opcji komunikatu

Nazwa	Struktura
MQGMO_STRUC_ID	"GMO↯"
MQGMO_STRUC_ID_ARRAY	'G','M','O','↯'

Uwaga: Symbol ↯ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQGMO_VERSION_1	1	X'00000001'
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
MQGMO_CURRENT_VERSION	4	X'00000004'

Opcje pobierania komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQGMO_WAIT	1	X'00000001'
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF QUIESCING	8192	X'00002000'
MQGMO_SYNCPOINT,	2	X'00000002'
MQGMO_SYNCPOINT_IF_PERSISTENT	4096	X'00001000'
MQGMO_NO_SYNCPOINT	4	X'00000004'
MQGMO_MARK_SKIP_BACKOUT	128	X'00000080'
MQGMO_BROWSE_FIRST	16	X'00000010'
MQGMO_BROWSE_NEXT	32	X'00000020'
MQGMO_BROWSE_MSG_UNDER_CURSOR	2048	X'00000800'
MQGMO_BROWSE_HANDLE	17825808	X'01100010'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQGMO_BROWSE_CO_OP	18874384	X'01200010'
MQGMO_MSG_UNDER_CURSOR	256	X'00000100'
Blokada MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
Komunikat MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_ * (Status grupy)

Nazwa	Wartość
MQGS_NOT_IN_GROUP	'¬'
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

MQHA_ * (selektory uchwytów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHA_FIRST	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_ * (uchwyty Bag-Bag Handles)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_ * (uchwyty połączeń)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_ * (uchwyt komunikatu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_ * (uchwyt obiektu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_ * (obsługa stanów formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHSTATE_INACTIVE,	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_ * (selektory atrybutów całkowitych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ADOPT_CONTEXT	260	X'00000104'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
MQIA_CLUSTER_OBJECT_STATE	256	X'00000100'
MQIA_CLUSTER_PUB_ROUTE	255	X'000000FF'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'
MQIA_DISPLAY_TYPE	262	X'00000106'
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'
MQIA_GROUP_UR	221	X'000000DD'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'
MQIA_INHIBIT_EVENT	48	X'00000030'
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_INHIBIT_SUB	182	X'000000B6'
MQIA_INTRA_GROUP_queuing	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	266	X'0000010A'
MQIA_LDAP_AUTHORMD	263	X'00000107'
MQIA_LDAP_NESTGRP	264	X'00000108'
MQIA_LDAP_SECURE_COMM	261	X'00000105'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'
MQIA_NPM_DELIVERY	196	X'000000C4'
MQIA_OPEN_INPUT_COUNT	17	X'00000011'
MQIA_OPEN_OUTPUT_COUNT	18	X'00000012'
MQIA_OUTBOUND_PORT_MAX	140	X'0000008C'
MQIA_OUTBOUND_PORT_MIN	110	X'0000006E'
MQIA_PAGESET_ID	62	X'0000003E'
MQIA_PERFORMANCE_EVENT	53	X'00000035'
MQIA_PLATFORM	32	X'00000020'
MQIA_PM_DELIVERY	195	X'000000C3'
MQIA_PROPERTY_CONTROL	190	X'000000BE'
MQIA_PROT_POLICY_CAPABILITY	251	X'000000FB'
MQIA_PROXY_SUB	199	X'000000C7'
MQIA_PUB_COUNT	215	X'000000D7'
MQIA_PUB_SCOPE	219	X'000000DB'
MQIA_PUBSUB_CLUSTER	249	X'000000F9'
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMGR_CFCONLOS	245	X'000000F5'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_QMOPT_CONS_ERROR_MSGS	153	X'00000099'
MQIA_QMOPT_CONS_INFO_MSGS	151	X'00000097'
MQIA_QMOPT_CONS_REORG_MSGS	156	X'0000009C'
MQIA_QMOPT_CONS_SYSTEM_MSGS	157	X'0000009D'
MQIA_QMOPT_CONS_WARNING_MSGS	152	X'00000098'
MQIA_QMOPT_CSMT_ON_ERROR	150	X'00000096'
MQIA_QMOPT_INTERNAL_DUMP	170	X'000000AA'
MQIA_QMOPT_LOG_COMMS_MSGS	162	X'000000A2'
MQIA_QMOPT_LOG_CRITICAL_MSGS	161	X'000000A1'
MQIA_QMOPT_LOG_ERROR_MSGS	160	X'000000A0'
MQIA_QMOPT_LOG_INFO_MSGS	158	X'0000009E'
MQIA_QMOPT_LOG_REORG_MSGS	163	X'000000A3'
MQIA_QMOPT_LOG_SYSTEM_MSGS	164	X'000000A4'
MQIA_QMOPT_LOG_WARNING_MSGS	159	X'0000009F'
MQIA_QMOPT_TRACE_COMMS	166	X'000000A6'
MQIA_QMOPT_TRACE_CONVERSION	168	X'000000A8'
MQIA_QMOPT_TRACE_REORG	167	X'000000A7'
MQIA_QMOPT_TRACE_MQI_CALLS	165	X'000000A5'
MQIA_QMOPT_TRACE_SYSTEM	169	X'000000A9'
MQIA_QSG_DISP	63	X'0000003F'
MQIA_READ_AHEAD	189	X'000000BD'
MQIA_RECEIVE_TIMEOUT	111	X'0000006F'
MQIA_RECEIVE_TIMEOUT_MIN	113	X'00000071'
MQIA_RECEIVE_TIMEOUT_TYPE	112	X'00000070'
MQIA_REMOTE_EVENT	50	X'00000032'
MQIA_RETENTION_INTERVAL	21	X'00000015'
MQIA_REVERSE_DNS_LOOKUP	254	X'000000FE'
MQIA_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'
MQIA_STATISTICS_INTERVAL	131	X'00000083'
MQIA_STATISTICS_MQI	127	X'0000007F'
MQIA_STATISTICS_Q	128	X'00000080'
MQIA_SUB_COUNT	204	X'000000CC'
MQIA_SUB_SCOPE	218	X'000000DA'
MQIA_SYNCPOINT	30	X'0000001E'
MQIA_TCP_CHANNELS	114	X'00000072'
MQIA_TCP_KEEP_ALIVE	115	X'00000073'
MQIA_TCP_STACK_TYPE	116	X'00000074'
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
MQIA_TOPIC_NODE_COUNT	253	X'000000FD'
MQIA_TOPIC_TYPE	208	X'000000D0'
MQIA_TRACE_ROUTE_RECORDING	137	X'00000089'
MQIA_TREE_LIFE_TIME	183	X'000000B7'
MQIA_TRIGGER_CONTROL	24	X'00000018'
MQIA_TRIGGER_DEPTH	29	X'0000001D'
MQIA_TRIGGER_INTERVAL	25	X'00000019'
MQIA_TRIGGER_MSG_PRIORITY	26	X'0000001A'
MQIA_TRIGGER_TYPE	28	X'0000001C'
MQIA_TRIGGER_RESTART	91	X'0000005B'
MQIA_USAGE	12	X'0000000C'
MQIA_USE_DEAD_LETTER_Q	234	X'000000EA'
MQIA_USER_LIST	2000	X'000007D0'
MQIA_WILDCARD_OPERATION	216	X'000000D8'
MQIA_XR_CAPABILITY	243	X'000000F3'

MQIACF_* (typy parametrów całkowitoliczbowych w formacie komendy)

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

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
TRYB MQIACF_MODE	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'
MQIACF_EVENT_APPL_TYPE	1010	X'000003F2'
MQIACF_EVENT_ORIGIN	1011	X'000003F3'
MQIACF_PARAMETER_ID,	1012	X'000003F4'
MQIACF_ERROR_ID	1013	X'000003F5'
MQIACF_ERROR_IDENTIFIER	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
MQIACF_OBJECT_TYPE	1016	X'000003F8'
TYP_MQIACF_ESCAPE_TYPE	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
Kwalifikator MQIACF_REASON_QUALIFIER	1020	X'000003FC'
MQIACF_COMMAND	1021	X'000003FD'
OPCJE MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_OPEN_TYPE	1023	X'000003FF'
ID_PROCESU 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'
Kod_przyczyny MQIACF_CONV_REASON_CODE	1072	X'00000430'
MQIACF_BRIDGE_TYPE	1073	X'00000431'
ZAPYTANIE_MQIACF_ZAPYTANIE	1074	X'00000432'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_WAIT_INTERVAL	1075	X'00000433'
OPCJE 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'
OPCJE 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'
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'
OPCJE MQIACF_DELETE_OPTIONS	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
MQIACF_REFRESH_INTERVAL	1094	X'00000446'
MQIACF_REFRESH_REPOSITORY	1095	X'00000447'
MQIACF_REMOVE_QUEUES	1096	X'00000448'
MQIACF_OPEN_INPUT_TYPE	1098	X'0000044A'
MQIACF_OPEN_OUTPUT	1099	X'0000044B'
MQIACF_OPEN_SET	1100	X'0000044C'
MQIACF_OPEN_INQUIRE	1101	X'0000044D'
MQIACF_OPEN_BROWSE	1102	X'0000044E'
MQIACF_Q_STATUS_TYPE	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
MQIACF_Q_STATUS	1105	X'00000451'
MQIACF_SECURITY_TYPE	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
OPCJE MQIACF_CONNECT_OPTIONS	1108	X'00000454'
MQIACF_CONN_INFO_TYPE	1110	X'00000456'
MQIACF_CONN_INFO_CONN	1111	X'00000457'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
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'
TYP_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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_SECURITY_INTERVAL	1153	X'00000481'
MQIACF_SECURITY_SWITCH	1154	X'00000482'
MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
MQIACF_USAGE_TYPE	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
MQIACF_USAGE_PERSIST_PAGES	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
MQIACF_PAGESET_STATUS	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
MQIACF_USAGE_DATA_SET_TYPE	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'
MQIACF_MOVE_COUNT	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURATION_OBJECTS	1173	X'00000495'
MQIACF_CONFIGURATION_EVENTS,	1174	X'00000496'
MQIACF_SYSP_TYPE	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVE	1177	X'00000499'
MQIACF_SYSP_MAX_READ_TAPES	1178	X'0000049A'
MQIACF_SYSP_IN_BUFFER_SIZE	1179	X'0000049B'
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
MQIACF_SYSP_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVE	1183	X'0000049F'
MQIACF_SYSP_DUAL_ARCHIVE	1184	X'000004A0'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_SYSP_DUAL_BSDS	1185	X'000004A1'
MQIACF_SYSP_MAX_CONNS	1186	X'000004A2'
MQIACF_SYSP_MAX_CONNS_TIME	1187	X'000004A3'
MQIACF_SYSP_MAX_CONNS_BACK	1188	X'000004A4'
MQIACF_SYSP_EXIT_INTERVAL	1189	X'000004A5'
MQIACF_SYSP_EXIT_TASKS	1190	X'000004A6'
MQIACF_SYSP_CHKPOINT_COUNT	1191	X'000004A7'
MQIACF_SYSP_OTMA_INTERVAL	1192	X'000004A8'
MQIACF_SYSP_Q_INDEX_DEFER	1193	X'000004A9'
MQIACF_SYSP_DB2_TASKS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'
MQIACF_SYSP_ARCHIVE_WTOR	1205	X'000004B5'
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'
MQIACF_SYSP_ALLOC_SECONDARY	1210	X'000004BA'
MQIACF_SYSP_PROTECT	1211	X'000004BB'
MQIACF_SYSP_QUIESCE_INTERVAL	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
MQIACF_SYSP_UNIT_ADDRESS	1214	X'000004BE'
MQIACF_SYSP_UNIT_STATUS	1215	X'000004BF'
MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
MQIACF_SYSP_OFFLOAD_STATUS	1219	X'000004C3'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_SYSP_TOTAL_LOGS	1220	X'000004C4'
MQIACF_SYSP_FULL_LOGS	1221	X'000004C5'
MQIACF_LISTENER_ATTRS	1222	X'000004C6'
MQIACF_LISTENER_STATUS_ATTRS	1223	X'000004C7'
MQIACF_SERVICE_ATTRS	1224	X'000004C8'
MQIACF_SERVICE_STATUS_ATTRS	1225	X'000004C9'
MQIACF_Q_TIME_INDICATOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
OPCJE MQIACF_AUTH_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'
STATUS MQIACF_CHINIT_STATUS	1232	X'000004D0'
MQIACF_CMD_SERVER_STATUS	1233	X'000004D1'
MQIACF_ROUTE_DETAIL	1234	X'000004D2'
MQIACF_RECORDED_ACTIVITIES	1235	X'000004D3'
MQIACF_MAX_ACTIVITIES	1236	X'000004D4'
MQIACF_DISCONTINUITY_COUNT	1237	X'000004D5'
MQIACF_ROUTE_AKUMULACJA	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_WAŻNOŚCI	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'
MQIACF_ORIGINAL_LENGTH	1251	X'000004E3'
MQIACF_PERSISTENCE,	1252	X'000004E4'
MQIACF_PRIORITY	1253	X'000004E5'
MQIACF_REASON_CODE	1254	X'000004E6'
RAPORT MQIACF_REPORT	1255	X'000004E7'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_VERSION	1256	X'000004E8'
MQIACF_UNRECORDED_ACTIVITIES	1257	X'000004E9'
MQIACF_MONITORING	1258	X'000004EA'
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
MQIACF_SERVICE_STATUS	1260	X'000004EC'
MQIACF_Q_TYPES	1261	X'000004ED'
MQIACF_USER_ID_SUPPORT	1262	X'000004EE'
MQIACF_INTERFACE_VERSION	1263	X'000004EF'
MQIACF_AUTH_SERVICE_ATTRS	1264	X'000004F0'
MQIACF_USAGE_EXPAND_TYPE	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_DB2_BLOB_TASKS	1267	X'000004F3'
MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
MQIACF_TOPIC_ATTRS	1269	X'000004F5'
MQIACF_PUBSUB_PROPERTIES	1271	X'000004F7'
MQIACF_DESTINATION_CLASS	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SUBSCRIPTION_SCOPE	1275	X'000004FB'
MQIACF_VARIABLE_USER_ID	1277	X'000004FD'
MQIACF_REQUEST_ONLY	1280	X'00000500'
MQIACF_PUB_PRIORITY	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
MQIACF_WILDCARD_SCHEMA	1288	X'00000508'
MQIACF_SUB_TYPE	1289	X'00000509'
MQIACF_MESSAGE_COUNT	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSION	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
STATUS MQIACF_TOPIC_STATUS	1295	X'0000050F'
MQIACF_TOPIC_SUB	1296	X'00000510'
MQIACF_TOPIC_PUB	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_TOPIC_STATUS_ATTRS	1301	X'00000515'
MQIACF_TOPIC_STATUS_TYPE,	1302	X'00000516'
OPCJE MQIACF_SUB_OPTIONS	1303	X'00000517'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_CLEAR_TYPE	1305	X'00000519'
MQIACF_CLEAR_SCOPE	1306	X'0000051A'
MQIACF_SUB_LEVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE,	1308	X'0000051C'
MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
MQIACF_SELECTOR_TYPE	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_CHLAUTH_TYPE	1352	X'00000548'
MQXR_DIAGNOSTICS_TYPE	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
MQIACF_OPERATION_ID	1356	X'0000054C'
MQIACF_API_CALLER_TYPE,	1357	X'0000054D'
MQIACF_API_ENVIRONMENT	1358	X'0000054E'
MQIACF_TRACE_DETAIL	1359	X'0000054F'
MQIACF_HOBJ	1360	X'00000550'
MQIACF_CALL_TYPE	1361	X'00000551'
MQIACF_MQCB_OPERATION	1362	X'00000552'
MQIACF_MQCB_TYPE	1363	X'00000553'
OPCJE MQIACF_MQCB_OPTIONS	1364	X'00000554'
MQIACF_CLOSE_OPTIONS	1365	X'00000555'
MQIACF_CTL_OPERATION	1366	X'00000556'
OPCJE MQIACF_GET_OPTIONS	1367	X'00000557'
MQIACF_RECS_PRESENT	1368	X'00000558'
MQIACF_KNOWN_DEST_COUNT	1369	X'00000559'
MQIACF_UNKNOWN_DEST_COUNT	1370	X'0000055A'
MQIACF_INVALID_DEST_COUNT	1371	X'0000055B'
MQIACF_RESOLVED_TYPE	1372	X'0000055C'
OPCJE MQIACF_PUT_OPTIONS	1373	X'0000055D'
MQIACF_BUFFER_LENGTH	1374	X'0000055E'
MQIACF_TRACE_DATA_LENGTH	1375	X'0000055F'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_SMDS_EXPANDST	1376	X'00000560'
MQIACF_STRUC_LENGTH	1377	X'00000561'
MQIACF_ITEM_COUNT	1378	X'00000562'
MQIACF_EXPIRY_TIME	1379	X'00000563'
CZAS_POŁĄCZENIA MQIACF_CONNECT_TIME	1380	X'00000564'
MQIACF_DISCONNECT_TIME	1381	X'00000565'
MQIACF_HSUB	1382	X'00000566'
OPCJE MQIACF_SUBRQ_OPTIONS	1383	X'00000567'
MQIACF_XA_RMID	1384	X'00000568'
MQIACF_XA_FLAGS,	1385	X'00000569'
MQIACF_XA_RETCODE	1386	X'0000056A'
MQIACF_XA_HANDLE	1387	X'0000056B'
MQIACF_XA_RETVAL	1388	X'0000056C'
MQIACF_STATUS_TYPE	1389	X'0000056D'
MQIACF_XA_COUNT	1390	X'0000056E'
MQIACF_SELECTOR_COUNT	1391	X'0000056F'
MQIACF_SELECTORS	1392	X'00000570'
MQIACF_INTATTR_COUNT	1393	X'00000571'
MQIACF_INTATTRS	1394	X'00000572'
MQIACF_SUBRQ_ACTION	1395	X'00000573'
MQIACF_NUM_PUBS	1396	X'00000574'
MQIACF_POINTER_SIZE	1397	X'00000575'
MQIACF_REMOVE_AUTHREC	1398	X'00000576'
MQIACF_XR_ATTRS	1399	X'00000577'
MQIACF_APPL_FUNCTION_TYPE	1400	X'00000578'
MQIACF_AMQP_ATTRS	1401	X'00000579'
MQIACF_EXPORT_TYPE	1402	X'0000057A'
MQIACF_EXPORT_ATTRS	1403	X'0000057B'
MQIACF_SYSTEM_OBJECTS	1404	X'0000057C'
MQIACF_CONNECTION_SWAP	1405	X'0000057D'
MQIACF_AMQP_DIAGNOSTICS_TYPE	1406	X'0000057E'
MQIACF_BUFFER_POOL_LOCATION	1408	X'00000580'
MQIACF_LDAP_CONNECTION_STATUS	1409	X'00000581'
MQIACF_SYSP_MAX_ACE_POOL	1410	X'00000582'
MQIACF_PAGECLAS	1411	X'00000583'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
▶ V8.0.0.4 ▶ V8.0.0.4 MQIACF_AUTH_REC_TYPE	1412	X'00000584'
MQIACF_LAST_USED	1412	X'00000584'

MQIACH_* (typy liczb całkowitych w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'
MQIACH_DISC_INTERVAL	1503	X'000005DF'
MQIACH_SHORT_TIMER	1504	X'000005E0'
MQIACH_SHORT_RETRY	1505	X'000005E1'
MQIACH_LONG_TIMER	1506	X'000005E2'
MQIACH_LONG_RETRY	1507	X'000005E3'
MQIACH_PUT_AUTHORITY	1508	X'000005E4'
MQIACH_SEQUENCE_NUMBER_WRAP	1509	X'000005E5'
MQIACH_MAX_MSG_LENGTH	1510	X'000005E6'
MQIACH_CHANNEL_TYPE	1511	X'000005E7'
MQIACH_DATA_COUNT	1512	X'000005E8'
MQIACH_NAME_COUNT	1513	X'000005E9'
MQIACH_MSG_SEQUENCE_NUMBER	1514	X'000005EA'
MQIACH_DATA_CONVERSION	1515	X'000005EB'
MQIACH_IN_DOUBT	1516	X'000005EC'
MQIACH_MCA_TYPE	1517	X'000005ED'
MQIACH_SESSION_COUNT	1518	X'000005EE'
MQIACH_ADAPTER	1519	X'000005EF'
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'
MQIACH_NPM_SPEED	1562	X'0000061A'
MQIACH_HB_INTERVAL	1563	X'0000061B'
MQIACH_BATCH_INTERVAL	1564	X'0000061C'
MQIACH_NETWORK_PRIORITY	1565	X'0000061D'
MQIACH_KEEP_ALIVE_INTERVAL	1566	X'0000061E'
MQIACH_BATCH_HB	1567	X'0000061F'
MQIACH_SSL_CLIENT_AUTH	1568	X'00000620'
MQIACH_ALLOC_RETRY	1570	X'00000622'
MQIACH_ALLOC_FAST_TIMER	1571	X'00000623'
MQIACH_ALLOC_SLOW_TIMER	1572	X'00000624'
MQIACH_DISC_RETRY	1573	X'00000625'
MQIACH_PORT_NUMBER	1574	X'00000626'
MQIACH_HDR_COMPRESSION	1575	X'00000627'
MQIACH_MSG_COMPRESSION	1576	X'00000628'
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'
MQIACH_BACKLOG	1602	X'00000642'
MQIACH_XMITQ_TIME_INDICATOR	1604	X'00000644'
MQIACH_NETWORK_TIME_INDICATOR	1605	X'00000645'
MQIACH_EXIT_TIME_INDICATOR	1606	X'00000646'
MQIACH_BATCH_SIZE_INDICATOR	1607	X'00000647'
MQIACH_XMITQ_MSGS_AVAILABLE	1608	X'00000648'
MQIACH_CHANNEL_SUBSTATE	1609	X'00000649'
MQIACH_SSL_KEY_RESETS	1610	X'0000064A'
MQIACH_COMPRESSION_RATE	1611	X'0000064B'
MQIACH_COMPRESSION_TIME	1612	X'0000064C'
MQIACH_MAX_XMIT_SIZE	1613	X'0000064D'
MQIACH_DEF_CHANNEL_DISP	1614	X'0000064E'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_PROTOCOL	1643	X'0000066B'
MQIACH_AMQPKEEPALIVE	1644	X'0000066C'
MQIACH_SECURITY_PROTOCOL	1645	X'0000066D'
MQIACH_LAST_USED	1645	X'0000066D'

MQIAMO_* (typy parametrów monitorowania liczby całkowitej w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_FIRST	701	X'000002BD'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'
MQIAMO_BACKOUTS,	704	X'000002C0'
MQIAMO_BROWSES	705	X'000002C1'
MQIAMO_BROWSE_MAX_BYTES	706	X'000002C2'
MQIAMO_BROWSE_MIN_BYTES (min_bajty)	707	X'000002C3'
MQIAMO_BROWSES_FAILED	708	X'000002C4'
MQIAMO_ZAMYKA	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_DYSKI	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_PARTIE	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
MQIAMO_POBIERA	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_PARTIE	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_OTWIERA	733	X'000002DD'
MQIAMO_PUT1S	734	X'000002DE'
MQIAMO_PUTS	735	X'000002DF'
MQIAMO_PUT_MAX_BYTES	736	X'000002E0'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_PUT_MIN_BYTES	737	X'000002E1'
MQIAMO_PUT_RETRIES,	738	X'000002E2'
MQIAMO_Q_MAX_DEPTH	739	X'000002E3'
MQIAMO_Q_MIN_DEPTH	740	X'000002E4'
MQIAMO_Q_TIME_AVG	741	X'000002E5'
MQIAMO_Q_TIME_MAX	742	X'000002E6'
MQIAMO_Q_TIME_MIN	743	X'000002E7'
MQIAMO_SETS	744	X'000002E8'
MQIAMO_CONNS_FAILED	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_FAILED	752	X'000002F0'
MQIAMO_SETS_FAILED	753	X'000002F1'
MQIAMO_PUTS_FAILED	754	X'000002F2'
MQIAMO_PUT1S_FAILED	755	X'000002F3'
Niepowodzenie MQIAMO_CLOSSES_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 (nie powiodło się)	780	X'0000030C'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_TOPIC_PUT1S	781	X'0000030D'
MQIAMO_TOPIC_PUT1S_FAILED	782	X'0000030E'
MQIAMO_PUBLISH_MSG_COUNT	784	X'00000310'
MQIAMO_UNSUBS_DUR	786	X'00000312'
MQIAMO_UNSUBS_NDUR	787	X'00000313'
MQIAMO_UNSUBS_FAILED	788	X'00000314'
MQIAMO_INTERVAL	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'
MQIAMO_FEEDBACK_MODE,	793	X'00000319'
MQIAMO_RELIABILITY_TYPE	794	X'0000031A'
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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MSGS_DOSTARCZONEGO	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROPPED	822	X'00000336'
MQIAMO_PKTS_ZDUPLIKOWANE	823	X'00000337'
MQIAMO_NACKS_CREATED	824	X'00000338'
MQIAMO_NACK_PKTS_SENT	825	X'00000339'
MQIAMO_REPAIR_PKTS_RQSTD	826	X'0000033A'
MQIAMO_REPAIR_PKTS_RCVD	827	X'0000033B'
MQIAMO_PKTS_REPAIRED	828	X'0000033C'
MQIAMO_TOTAL_MSGS_RCVD	829	X'0000033D'
MQIAMO_TOTAL_MSGS_BYTES_RCVD	830	X'0000033E'
MQIAMO_TOTAL_REPAIR_PKTS_RCVD	831	X'0000033F'
MQIAMO_TOTAL_REPAIR_PKTS_RQSTD	832	X'00000340'
MQIAMO_TOTAL_MSGS_PROCESSED	833	X'00000341'
MQIAMO_TOTAL_MSGS_SELECTED,	834	X'00000342'
MQIAMO_TOTAL_MSGS_EXPIRED	835	X'00000343'
MQIAMO_TOTAL_MSGS_DOSTARCZONEGO	836	X'00000344'
MQIAMO_TOTAL_MSGS_ZWRÓCONE	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

MQIAMO64_* (format komend-64-bitowe typy parametrów monitorowania liczb całkowitych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO64_AVG_Q_TIME	703	X'000002BF'
MQIAMO64_Q_TIME_AVG	741	X'000002E5'
MQIAMO64_Q_TIME_MAX	742	X'000002E6'
MQIAMO64_Q_TIME_MIN	743	X'000002E7'
MQIAMO64_BROWSE_BYTES	745	X'000002E9'
MQIAMO64_BYTES	746	X'000002EA'
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_* (selektory systemowe-liczba całkowita)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIASY_FIRST	-1	X'FFFFFFFF'
MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
TYP_MQIASY_MQ	-2	X'FFFFFFFE'
MQIASY_COMMAND,	-3	X'FFFFFFFD'
MQIASY_MSG_SEQ_NUMBER	-4	X'FFFFFFFC'
KONTROLA MQIASY_CONTROL	-5	X'FFFFFFFB'
KMQIASY_KOD_KOI	-6	X'FFFFFFFA'
Przyczyna MQIASY_PRZYCZYNA	-7	X'FFFFFFF9'
OPCJE MQIASY_BAG_OPTIONS	-8	X'FFFFFFF8'
MQIASY_VERSION	-9	X'FFFFFFF7'
MQIASY_LAST_USED	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_* (uwierzytelniający nagłówek informacji IMS)

Nazwa	Wartość
MQIAUT_NONE	"~~~~~"
MQIAUT_NONE_ARRAY	'~','~','~','~','~','~','~','~','~'

Uwaga: Symbol ~ reprezentuje pojedynczy pusty znak.

MQIAV_* (wartości atrybutów całkowitych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
NIE DOTYCZY MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_UNDEFINED	-2	X'FFFFFFFE'

MQICM_* (IMS -Tryby zatwierdzania nagłówka informacji)

Nazwa	Wartość
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_* (opcje wątpliwych formatów komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIDO_COMMIT	1	X'00000001'
MQIDO_BACKOUT	2	X'00000002'

MQIEP_* (Punkty wejścia interfejsu)

Struktura parametrów zabezpieczeń połączenia

Nazwa	Struktura
MQIEP_STRUC_ID	"IEP¬"
MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_* (kolejka intra-grupa kolejkowania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIGQ_WYŁĄCZONE	0	X'00000000'
MQIGQ_ENABLED	1	X'00000001'

MQIGQPA_* (uprawnienie do umieszczania w kolejkach w kolejce Intra-Group)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIGQPA_DEFAULT	1	X'00000001'
MQIGQPA_CONTEXT	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

MQIIH_* (struktura nagłówka informacji IMS i flagi)

Struktura nagłówka informacyjnego produktu IMS

Nazwa	Struktura
MQIIH_STRUC_ID	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

Flagi nagłówek informacji IMS

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (Zapytaj o opcje i strukturę właściwości komunikatu)

Sprawdź strukturę opcji właściwości komunikatu

Nazwa	Struktura
MQIMPO_STRUC_ID,	"IMPO"
MQIMPO_STRUC_ID_ARRAY	'I', 'M', 'P', 'O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIMPO_VERSION_1	1	X'00000001'
MQIMPO_CURRENT_VERSION	1	X'00000001'

Zapytaj o opcje właściwości komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
TYP_KONTEKSTU 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'
WARTOŚĆ MQIMPO_CONVERT_VALUE	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_* (Przychodzące dyspozycje w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQINBD_Q_MGR	0	X'00000000'
MQINBD_GROUP	3	X'00000003'

MQIND_* (wartości indeksu specjalnego)

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

MQIPADDR_* (wersje adresów IP)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIPADDR_IPv4	0	X'00000000'
MQIPADDR_IPv6	1	X'00000001'

MQISS_* (zasięgi zabezpieczeń nagłówka informacyjnego produktu IMS)

Nazwa	Wartość
SPRAWDZANIE MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_* (typy indeksów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIT_NONE	0	X'00000000'
MQIT_MSG_ID,	1	X'00000001'
ID_MQIT_CORREL_ID	2	X'00000002'
MQIT_MSG_TOKEN,	4	X'00000004'
MQIT_GROUP_ID	5	X'00000005'

MQITEM_* (typ elementu dla elementu mqInquireItemInfo)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (identyfikator instancji transakcji nagłówka informacji IMS)

Nazwa	Wartość
MQITII_NONE	X'00...00' (16 zer)

Nazwa	Wartość
MQITII_NONE_ARRAY	'\0', '\0', ... (16 zer)

MQITS_ * (stany transakcji nagłówka informacji IMS)

Nazwa	Wartość
MQITS_IN_CONVERSATION,	'C'
MQITS_NOT_IN_CONVERSATION	'¬'
MQITS_ARCHITECTED	'A'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

MQKAI_ * (przedział czasuKeepAlive)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_ * (administrowanie główne)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

MQMCAS_ * (Status agenta kanału komunikatów-Status)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMCAS_ZATRZYMANÝ	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_ * (typy MCA)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMCAT_PROCESS	1	X'00000001'
MQMCAT_THREAD	2	X'00000002'

MQMCD_ * (Informacje o znaczniku opcji publikowania/subskrypcji)

Opcje Publikowania/subskrypcji Znaczniki Deskryptora Treści Komunikatu (mcd)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMCD_FOLDER_WERSJA	1	X'00000001'

Nazwy znaczników znaczników opcji publikowania/subskrypcji

Nazwa	Wartość
MQMCD_MSG_DOMAIN	"Msd"

Nazwa	Wartość
MQMCD_MSG_SET	"Set"
MQMCD_MSG_TYPE,	"Type"
MQMCD_MSG_FORMAT	"Fmt"

Nazwy znaczników XML znaczników opcji publikowania/subskrypcji

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

Wartości znaczników znaczników opcji publikowania/subskrypcji

Nazwa	Wartość
MQMCD_DOMAIN_NONE (brak)	"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_* (struktura deskryptora komunikatu)

Nazwa	Struktura
MQMD_STRUC_ID	"MD↵"
Tabela MQMD_STRUC_ID_ARRAY	'M', 'D', '↵', '↵'

Uwaga: Symbol ↵ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_* (struktura rozszerzenia deskryptora komunikatu)

Nazwa	Struktura
MQMDE_STRUC_ID	"MDE↵"
MQMDE_STRUC_ID_ARRAY	'M','D','E','↵'

Uwaga: Symbol ↵ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMDE_VERSION_2	2	X'00000002'
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_* (Flagi rozszerzenia deskryptora komunikatu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMDEF_NONE	0	X'00000000'

MQMDS_* (sekwencja dostarczania komunikatów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMDS_PRIORITY,	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_* (flagi wiadomości)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMF_SEGMENTATION_INHIBITED	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

MQMHBO_* (uchwyt komunikatu do opcji buforu i struktury)

Uchwyt komunikatu do struktury opcji buforu

Nazwa	Struktura
MQMHBO_STRUC_ID	"MHBO"

Nazwa	Struktura
Tablica MQMHBO_STRUC_ID_ARRAY	'M', 'H', 'B', 'O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

Opcje Obsługi Komunikatów Do Buforu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_NONE	0	X'00000000'

MQMI_* (identyfikator komunikatu)

Nazwa	Wartość
MQMI_NONE	X'00...00' (24 znaki puste)
MQMI_NONE_ARRAY	'\0', '\0', ... (24 znaki puste)

MQMMBI_* (znacznik komunikatu-Odstęp czasu przeglądania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMMBI_UNLIMITED	-1	X'FFFFFFFF'

MQMO_* (opcje zgodności)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_KORELID	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_* (opcje trybu wiersza komend)

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

MQMON_* (wartości monitorowania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (typy komunikatów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
Raport_menedżera_mQMT	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_* (Token komunikatu)

Nazwa	Wartość
MQMTOK_BRAK	X'00...00' (16 zer)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 zer)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMTOK_BRAK	X'00...00'	(16 nulls)
MQMTOK_NONE_ARRAY	'\0','\0',...	(16 nulls)

MQNC_* (liczba nazw)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Nr_NAME_NAME_COUNT MQNC_MAX_NAME_	256	X'00000100'

MQNPM_* (Nietrwąta Klasa Komunikacji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_* (NonPersistent-Speeds Komunikacji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_* (typy listy nazw)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQNT_NONE	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
MQNT_AUTH_INFO	4	X'00000004'
MQNT_ALL	1001	X'000003E9'

MQNVS_* (Nazwy dla łańcucha nazwa/wartość)

Nazwa	Wartość
MQNVS_APPL_TYPE	"OPT_APP_GRP↵"
MQNVS_MSG_TYPE	"OPT_MSG_TYPE↵"

Uwaga: Symbol ↵ reprezentuje pojedynczy pusty znak.

MQOA_* (Limity dla selektorów dla atrybutów obiektu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOA_FIRST	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_* (struktura deskryptora obiektu)

Nazwa	Struktura
MQOD_STRUC_ID	"OD↵↵"
MQOD_STRUC_ID_ARRAY	'0','D','↵','↵'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

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

MQOII_* (identyfikator instancji obiektu)

Nazwa	Wartość
MQOII_NONE	X'00...00' (24 znaki puste)
MQOII_NONE_ARRAY	'\0','\0',... (24 znaki puste)

MQOL_* (długość oryginalna)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOL_NIEZDEFINIOWANY	-1	X'FFFFFFFF'

MQOM_* (przestarzałe opcje komunikatów produktu Db2 w grupie Inquire)

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

MQOO_* (opcje otwarcia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOO_BIND_AS_Q_DEF	0	X'00000000'
MQOO_READ_AHEAD_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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOO_PASS_ALL_CONTEXT	512	X'00000200'
MQOO_SET_IDENTITY_CONTEXT,	1024	X'00000400'
MQOO_SET_ALL_CONTEXT	2048	X'00000800'
MQOO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQOO_FAIL_IF QUIESCING	8192	X'00002000'
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NOT_FIXED	32768	X'00008000'
MQOO_CO_OP	131072	X'00020000'
MQOO_RESOLVE_LOCAL_TOPIC	262144	X'00040000'
MQOO_NO_READ_AHEAD	524288	X'00080000'
MQOO_READ_AHEAD	1048576	X'00100000'
MQOO_BIND_ON_GROUP	4194304	X'00400000'

MQOO_* (używany tylko w języku C++)

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

MQOP_* (kody operacji dla MQCTL i MQCB)

Kody operacji dla obiektu MQCTL

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOP_START,	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

Kody operacji dla bazy MQCB

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOP_REGISTER	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Kody operacji dla MQCTL i MQCB

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOP_SUSPEND	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_* (Wartości związane ze strukturą MQOPEN_PRIV)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_* (operacje działania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_UNKNOWN	0	X'00000000'
MQOPER_BROWSE	1	X'00000001'
MQOPER_DISCARD	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
RAPORT MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'
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_* (typy obiektów i typy obiektów rozszerzonych)

Typy obiektów

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOT_NONE	0	X'00000000'
Kolejka 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'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOT_TOPIC	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
MQOT_LISTENER	11	X'0000000B'
Usługa MQOT_SERVICE	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Typy obiektów rozszerzonych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOT_ALL	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODEL_Q	1003	X'000003EB'
MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTE_Q	1005	X'000003ED'
Kanał MQOT_SENDER_CHANNEL	1007	X'000003EF'
Kanał MQOT_SERVER_CHANNEL	1008	X'000003F0'
Kanał MQOT_REQUESTER_CHANNEL	1009	X'000003F1'
Kanał MQOT_RECEIVER_CHANNEL	1010	X'000003F2'
MQOT_CURRENT_CHANNEL,	1011	X'000003F3'
Kanał MQOT_SAVED_CHANNEL	1012	X'000003F4'
MQOT_SVRCONN_CHANNEL	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL	1014	X'000003F6'
MQOT_SHORT_CHANNEL	1015	X'000003F7'
MQOT_CHLAUTH	1016	X'000003F8'
MQOT_REMOTE_Q_MGR_NAME,	1017	X'000003F9'
Strategia MQOT_PROT_POLICY	1019	X'000003FB'
Kanał MQOT_TT_CHANNEL	1020	X'000003FC'
MQOT_AMQP_CHANNEL	1021	X'000003FD'
 V8.0.0.4  V8.0.0.4 MQOT_AUTH_REC	1022	X'000003FE'

MQPA_* (uprawnienie do umieszczania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPA_DEFAULT	1	X'00000001'
MQPA_CONTEXT	2	X'00000002'
MQPA_ONLY_MCA,	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_* (deskryptor właściwości, obsługa i kontekst)

Struktura deskryptora właściwości

Nazwa	Struktura
Identyfikator MQPD_STRUC_ID	"PD--"
Tabela MQPD_STRUC_ID_ARRAY	'P','D',' ',' '

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Opcje deskryptora właściwości

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPD_NONE	0	X'00000000'

Opcje obsługi właściwości

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPD_SUPPORT_OPTIONAL	1	X'00000001'
MQPD_SUPPORT_REQUIRED	1048576	X'00100000'
MQPD_SUPPORT_REQUIRED_IF_LOCAL	1024	X'00000400'
MQPD_REJECT_UNSUP_MASK	-1048576	X'FFF00000'
MQPD_ACCEPT_UNSUP_IF_XMIT_MASK	1047552	X'000FFC00'
Maska MQPD_ACCEPT_UNSUP_MASK	1023	X'000003FF'

Kontekst właściwości

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

MQPER_* (wartości Trwałości)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPER_PERSISTENCE_AS_PARENT	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
MQPER_PERSISTENT	1	X'00000001'
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCE_AS_TOPIC_DEF	2	X'00000002'

MQPL_* (Platformy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MVS MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
Z_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_APPLIANCE	28	X'0000001C'
MQPL_NATIVE	1	X'00000001'

MQPMO_* (Umieść opcje komunikatów i strukturę maski publikacji)

Struktura opcji umieszczania komunikatów

Nazwa	Struktura
MQPMO_STRUC_ID	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_CURRENT_VERSION	3	X'00000003'
MQPMO_CURRENT_LENGTH	(value differs by platform or version)	

Opcje umieszczania komunikatów

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
MQPMO_NEW_MSG_ID	64	X'00000040'
MQPMO_NEW_CORREL_ID	128	X'00000080'
MQPMO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQPMO_PASS_ALL_CONTEXT	512	X'00000200'
MQPMO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQPMO_SET_ALL_CONTEXT	2048	X'00000800'
MQPMO_ALTERNATE_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_BRAK	0	X'00000000'

Opcje umieszczania komunikatów dla maski publikacji

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_* (Umieść pola rekordu komunikatu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPMRF_MSG_ID,	1	X'00000001'
MQPMRF_CORREL_ID	2	X'00000002'
Identyfikator MQPMRF_GROUP_ID	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPMRF_ACCOUNTING_TOKEN,	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_ * (opcje czyszczenia formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPO_YES	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_ * (priorytet)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

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

Wartości sterujące właściwości kolejki i kanału

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
KOMPATYBILNA_MQPROP_KOMPATYBILNOŚCI	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Maksymalna długość właściwości

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_ * (Umieść wartości odpowiedzi)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_* (publikowanie/subskrypcja)

Format komend-Status publikowania/subskrypcji

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_ZATRZYMYWANIE	2	X'00000002'
STATUS_MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
BŁĄD MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_REFUSED	6	X'00000006'

Znaczniki publikowania/subskrypcji jako łańcuchy

MQPS_COMMAND	"MQPSCommand"
MQPS_COMP_CODE	"MQPSCompCode"
MQPS_CORREL_ID	"MQPSCorrelId"
MQPS_DELETE_OPTIONS	"MQPSDelOpts"
Identyfikator MQPS_ERROR_ID	"MQPSErrorId"
MQPS_ERROR_POS	"MQPSErrorPos"
MQPS_INTEGER_DATA	"MQPSIntData"
MQPS_PARAMETER_ID,	"MQSParmId"
OPCJE MQPS_PUBLICATION_OPTIONS	"MQSPubOpts"
MQPS_PUBLISH_TIMESTAMP	"MQSPubTime"
MQPS_Q_MGR_NAME	"MQPSQMgrName"
MQPS_Q_NAME	"MQPSQName"
MQPS_REASON	"MQPSReason"
MQPS_REASON_TEXT	"MQPSReasonText"
Opcje MQPS_REGISTRATION_OPTIONS	"MQPSRegOpts"
MQPS_SEQUENCE_NUMBER	"MQPSSeqNum"
MQPS_STREAM_NAME	"MQPSStreamName"
MQPS_STRING_DATA,	"MQPSStringData"
MQPS_SUBSCRIPTION_IDENTITY,	"MQPSSubIdentity"
MQPS_SUBSCRIPTION_NAME	"MQPSSubName"
MQPS_SUBSCRIPTION_USER_DATA	"MQPSSubUserData"
MQPS_TOPIC	"MQPSTopic"
ID_UŻYTKOWNIKA MQPS_USER_ID	"MQPSUserId"

Znaczniki publikowania/subskrybowania jako łańcuchy puste

MQPS_COMMAND_B	"~MQPSCommand~"
MQPS_COMP_CODE_B	"~MQPSCompCode~"
MQPS_CORREL_ID_B	"~MQPSCorrelId~"
MQPS_DELETE_OPTIONS_B	"~MQPSDelOpts~"
MQPS_ERROR_ID_B	"~MQPSErrorId~"
MQPS_ERROR_POS_B	"~MQPSErrorPos~"
MQPS_INTEGER_DATA_B	"~MQPSIntData~"
MQPS_PARAMETER_ID_B	"~MQPSParmId~"
MQPS_PUBLICATION_OPTIONS_B	"~MQPSPubOpts~"
MQPS_PUBLISH_TIMESTAMP_B	"~MQPSPubTime~"
MQPS_Q_MGR_NAME_B	"~MQPSQMgrName~"
MQPS_Q_NAME_B	"~MQPSQName~"
MQPS_REASON_B	"~MQPSReason~"
MQPS_REASON_TEXT_B	"~MQPSReasonText~"
MQPS_REGISTRATION_OPTIONS_B	"~MQPSRegOpts~"
MQPS_SEQUENCE_NUMBER_B	"~MQPSSeqNum~"
MQPS_STREAM_NAME_B	"~MQPSStreamName~"
MQPS_STRING_DATA_B	"~MQPSStringData~"
MQPS_SUBSCRIPTION_IDENTITY_B	"~MQPSSubIdentity~"
MQPS_SUBSCRIPTION_NAME_B	"~MQPSSubName~"
MQPS_SUBSCRIPTION_USER_DATA_B	"~MQPSSubUserData~"
MQPS_TOPIC_B	"~MQPSTopic~"
MQPS_USER_ID_B	"~MQPSUserId~"

Uwaga: Symbol ~ reprezentuje pojedynczy pusty znak.

Wartości znaczników komendy publikowania/subskrypcji jako łańcuchy

MQPS_DELETE_PUBLICATION	"DeletePub"
MQPS_DEREGISTER_PUBLISHER	"DeregPub"
Subskrybent MQPS_DEREGISTER_SUBSKRYBENTA	"DeregSub"
MQPS_PUBLISH	"Publish"
MQPS_REGISTER_PUBLISHER	"RegPub"
MQPS_REGISTER_SUBSKRYBENTA	"RegSub"
MQPS_REQUEST_UPDATE,	"ReqUpdate"

Wartości znaczników komendy publikowania/subskrypcji jako łańcuchy puste

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

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Wartości znaczników opcji publikowania/subskrypcji jako łańcuchy

MQPS_ADD_NAME	"AddName"
MQPS_ANONYMOUS	"Anon"
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPS_DEREGISTER_ALL	"DeregAll"
MQPS_DIRECT_REQUESTS	"DirectReq"
MQPS_DUPLICATES_OK	"DupsOK"
MQPS_FULL_RESPONSE	"FullResp"
MQPS_INCLUDE_STREAM_NAME	"InclStreamName"
MQPS_INFORM_IF_ZACHOWANE	"InformIfRet"
MQPS_IS_RETAINED_PUBLICATION	"IsRetainedPub"
MQPS_JOIN_EXCLUSIVE	"JoinExcl"
MQPS_JOIN_SHARED	"JoinShared"
MQPS_LEAVE_ONLY	"LeaveOnly"
MQPS_LOCAL	"Local"
MQPS_LOCKED	"Locked"
MQPS_NEW_PUBLICATIONS_ONLY	"NewPubsOnly"
MQPS_NO_ALTERATION	"NoAlter"
MQPS_NO_REGISTRATION	"NoReg"
MQPS_NON_PERSISTENT	"NonPers"
MQPS_NONE	"None"
MQPS_OTHER_SUBSCRIBERS_ONLY	"OtherSubsOnly"
MQPS_PERSISTENT	"Pers"
MQPS_PERSISTENT_AS_PUBLISH	"PersAsPub"
MQPS_PERSISTENT_AS_Q	"PersAsQueue"
MQPS_PUBLISH_ON_REQUEST_ONLY	"PubOnReqOnly"

MQPS_RETAIN_PUBLIKACJA	"RetainPub"
MQPS_VARIABLE_USER_ID,	"VariableUserId"

Wartości znaczników opcji publikowania/subskrypcji jako łańcuchy zamknięte puste

MQPS_ADD_NAME_B	"¬AddName¬"
MQPS_ANONYMOUS_B	"¬Anon¬"
MQPS_CORREL_ID_AS_IDENTITY_B	"¬CorrelAsId¬"
MQPS_DEREGISTER_ALL_B	"¬DeregAll¬"
MQPS_DIRECT_REQUESTS_B	"¬DirectReq¬"
MQPS_DUPLICATES_OK_B	"¬DupsOK¬"
MQPS_FULL_RESPONSE_B	"¬FullResp¬"
MQPS_INCLUDE_STREAM_NAME_B	"¬InclStreamName¬"
MQPS_INFORM_IF_RETAINED_B	"¬InformIfRet¬"
MQPS_IS_RETAINED_PUBLICATION_B	"¬IsRetainedPub¬"
MQPS_JOIN_EXCLUSIVE_B	"¬JoinExcl¬"
MQPS_JOIN_SHARED_B	"¬JoinShared¬"
MQPS_LEAVE_ONLY_B	"¬LeaveOnly¬"
MQPS_LOCAL_B	"¬Local¬"
MQPS_LOCKED_B	"¬Locked¬"
MQPS_NEW_PUBLICATIONS_ONLY_B	"¬NewPubsOnly¬"
MQPS_NO_ALTERATION_B	"¬NoAlter¬"
MQPS_NO_REGISTRATION_B	"¬NoReg¬"
MQPS_NON_PERSISTENT_B	"¬NonPers¬"
MQPS_NONE_B	"¬None¬"
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"¬OtherSubsOnly¬"
MQPS_PERSISTENT_B	"¬Pers¬"
MQPS_PERSISTENT_AS_PUBLISH_B	"¬PersAsPub¬"
MQPS_PERSISTENT_AS_Q_B	"¬PersAsQueue¬"
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"¬PubOnReqOnly¬"
MQPS_RETAIN_PUBLICATION_B	"¬RetainPub¬"
MQPS_VARIABLE_USER_ID_B	"¬VariableUserId¬"

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

MQPSC_* (opcje publikowania/subskrypcji znaczników folderu komend publikowania/subskrypcji folderu komend (psc))

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSC_FOLDER_VERSION	1	X'00000001'

MQPSC_* (nazwy znaczników opcji publikowania/subskrypcji)

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

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

MQPSC_COMMAND_B	"<Command>"
MQPSC_COMMAND_E	"</Command>"
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"
MQPSC_PUBLICATION_OPTION_B	"<PubOpt>"
MQPSC_PUBLICATION_OPTION_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E	"</DelOpt>"
MQPSC_TOPIC_B	"<Topic>"
MQPSC_TOPIC_E	"</Topic>"
MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"
MQPSC_FILTER_B	"<Filter>"
MQPSC_FILTER_E	"</Filter>"

MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NAME_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
MQPSC_SEQUENCE_NUMBER_B	"<SeqNum>"
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"
MQPSC_SUBSCRIPTION_NAME_B	"<SubName>"
MQPSC_SUBSCRIPTION_NAME_E	"</SubName>"
MQPSC_SUBSCRIPTION_IDENTITY_B	"<SubIdentity>"
MQPSC_SUBSCRIPTION_IDENTITY_E	"</SubIdentity>"
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"
MQPSC_SUBSCRIPTION_USER_DATA_E	"</SubUserData>"
MQPSC_CORREL_ID_B	"<CorrelId>"
MQPSC_CORREL_ID_E	"</CorrelId>"

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

MQPSC_DELETE_PUBLICATION	"DeletePub"
Subskrybent MQPSC_DEREGISTER_SUBSKRYBENTA	"DeregSub"
MQPSC_PUBLISH	"Publish"
MQPSC_REGISTER_SUBSKRYBENTA	"RegSub"
MQPSC_REQUEST_UPDATE	"ReqUpdate"

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

MQPSC_NAZWA_ADD_NAME	"AddName"
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPSC_DEREGISTER_ALL	"DeregAll"
MQPSC_DUPLICATES_OK	"DupsOK"
MQPSC_FULL_RESPONSE	"FullResp"
MQPSC_INFORM_IF_ZACHOWANE	"InformIfRet"
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"
MQPSC_JOIN_SHARED	"JoinShared"
MQPSC_JOIN_EXCLUSIVE	"JoinExcl"
MQPSC_LEAVE_ONLY	"LeaveOnly"
MQPSC_LOCAL	"Local"
MQPSC_LOCKED	"Locked"

MQPSC_NEW_PUBS_ONLY	"NewPubsOnly"
MQPSC_NO_ALTERATION	"NoAlter"
MQPSC_NON_PERSISTENT	"NonPers"
MQPSC_OTHER_SUBS_ONLY	"OtherSubsOnly"
MQPSC_PERSISTENT	"Pers"
MQPSC_PERSISTENT_AS_PUBLISH	"PersAsPub"
MQPSC_PERSISTENT_AS_Q	"PersAsQueue"
MQPSC_NONE	"None"
MQPSC_PUB_ON_REQUEST_ONLY	"PubOnReqOnly"
MQPSC_RETAIN_PUB	"RetainPub"
MQPSC_VARIABLE_USER_ID	"VariableUserId"

MQPSCR_* (opcje publikowania/subskrypcji)

Znaczniki opcji publikowania/subskrypcji znaczników publikowania/subskrypcji (pscr)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSCR_FOLDER_VERSION	1	X'00000001'

Nazwy znaczników znaczników opcji publikowania/subskrypcji

MQPSCR_COMPLETION	"Completion"
MQPSCR_RESPONSE	"Response"
MQPSCR_REASON	"Reason"

Nazwy znaczników XML znaczników opcji publikowania/subskrypcji

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

Wartości znaczników znaczników opcji publikowania/subskrypcji

MQPSCR_OK	"ok"
MQPSCR_OSTRZEŻENIE	"warning"
Błąd MQPSCR_ERROR	"error"

MQPSM_* (tryb publikowania/subskrypcji)

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

MQPSPROP_* (Właściwości Komunikacji/subskrypcji)

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

MQPSST_* (typ statusu publikowania/subskrypcji w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSST_ALL	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PARENT,	2	X'00000002'
MQPSST_CHILD	3	X'00000003'

MQPUBO_* (opcje publikacji publikowania/subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLICATION	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRATION	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLIKACJA	16	X'00000010'

MQXPX_* (struktura parametru wyjścia routingu Publish/subscribe)

Nazwa	Struktura
MQXPX_STRUC_ID	"PXP¬"
MQXPX_STRUC_ID_ARRAY	'P','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXPX_VERSION_1	1	X'00000001'
MQXPX_CURRENT_VERSION	1	X'00000001'

MQQA_* (atrybuty kolejki)

Zablokuj pobieranie wartości

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

Zablokuj wartości wstawiane

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQA_PUT_INHIBITED	1	X'00000001'
MQQA_PUT_ALLOWED	0	X'00000000'

Współużytkowność kolejki

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Hartowanie wsteczne

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQA_BACKOUT_HARTOWANA	1	X'00000001'
MQQA_BACKOUT_NOT_HARTOWANE	0	X'00000000'

MQQDT_* (typy definicji kolejki)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQDT_PREDEFINIOWANE	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_* (flagi kolejki)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

MQQMDT_* (typy definicji menedżera kolejek w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Nadawca MQQMDT_EXPLICIT_CLUSTER_SENDER	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Nadawca MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
Nadawca MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
MQQMDT_CLUSTER_RECEIVER	3	X'00000003'

MQQMF_* (flagi menedżera kolejek)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

MQQMFC_* (funkcja menedżera kolejek w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQMFC_IMS_BRIDGE	1	X'00000001'
MQQMFC_DB2	2	X'00000002'

MQQMSTA_* (Status menedżera kolejek w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQMSTA_STARTING	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA QUIESCING	3	X'00000003'

MQQMT_* (typy menedżera kolejek w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQMT_NORMAL	0	X'00000000'
Repozytorium MQQMT_REPOSITORY	1	X'00000001'

MQQO_* (Opcje wyciszania formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQO_YES	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQSGD_* (Dyspozycje Grupy Współużytkowania Kolejek)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSGD_SHARED	2	X'00000002'
MQQSGD_GROUP	3	X'00000003'
MQQSGD_PRIVATE	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_* (format komendy Status QSG)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSGS_UNKNOWN	0	X'00000000'
MQQSGS_CREATED	1	X'00000001'
MQQSGS_ACTIVE	2	X'00000002'
MQQSGS_INACTIVE	3	X'00000003'
Funkcja MQQSGS_FAILED	4	X'00000004'
MQQSGS_PENDING	5	X'00000005'

MQQSIE_* (usługa kolejki formatu komendy-Zdarzenia przedziału czasu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSIE_NONE	0	X'00000000'
MQQSIE_WYSOKI	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_* (opcje otwarcia statusu kolejki w formacie komend dla SET, BROWSE, INPUT)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSO_NO	0	X'00000000'
MQQSO_YES	1	X'00000001'
MQQSO_SHARED	1	X'00000001'
MQQSO_EXCLUSIVE	2	X'00000002'

MQQSOT_* (typy otwierania statusu kolejki w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSOT_ALL	1	X'00000001'
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

MQQSUM_* (liczba niezatwierdzonych komunikatów statusu kolejki formatu komend)

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

MQQT_* (typy kolejek i rozszerzone typy kolejek)

Typy kolejek

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

Rozszerzone typy kolejek

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

MQRC_* (kody przyczyny)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
POŁĄCZONO MQRC_ALREADY_CONNECTED	2002	X'000007D2'
MQRC_BACKED_OUT	2003	X'000007D3'
MQRC_BUFFER_ERROR-BŁĄD	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'
Błąd MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
Błąd MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
Błąd MQRC_ENVIRONMENT_ERROR	2012	X'000007DC'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRC_EXPIRY_ERROR	2013	X'000007DD'
Błąd MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
BŁĄD MQRC_HCONN_ERROR	2018	X'000007E2'
BŁĄD MQRC_HOBJ_ERROR	2019	X'000007E3'
Błąd MQRC_INHIBIT_VALUE_ERROR	2020	X'000007E4'
MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_REACHED	2025	X'000007E9'
Błąd MQRC_MD_ERROR	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q,	2027	X'000007EB'
MQRC_MSG_TYPE_ERROR (BŁĄD)	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'
BŁĄD MQRC_OD_ERROR	2044	X'000007FC'
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
BŁĄD MQRC_OPTIONS_ERROR	2046	X'000007FE'
Błąd MQRC_PERSISTENCE_ERROR	2047	X'000007FF'
MQRC_PERSISTENT_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_PRZEKRACZA_MAKSIMUM	2049	X'00000801'
MQRC_PRIORITY_ERROR	2050	X'00000802'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_PUT_INHIBITED	2051	X'00000803'
MQRC_Q_DELETED	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'
MQRC_Q_TYPE_ERROR	2057	X'00000809'
Błąd MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
MQRC_REPORT_OPTIONS_ERROR,	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
MQRC_SECURITY_ERROR,	2063	X'0000080F'
MQRC_SELECTOR_COUNT_ERROR,	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
MQRC_SELECTOR_ERROR,	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TYPE	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NOT_AVAILABLE	2071	X'00000817'
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
MQRC_TRIGGER_CONTROL_ERROR	2075	X'0000081B'
MQRC_TRIGGER_DEPTH_ERROR	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
MQRC_TRIGGER_TYPE_ERROR	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
Funkcja MQRC_TRUNCATED_MSG_FAILED	2080	X'00000820'
MQRC_UNKNOWN_ALIAS_BASE_Q	2082	X'00000822'
MQRC_UNKNOWN_OBJECT_NAME	2085	X'00000825'
MQRC_UNKNOWN_OBJECT_Q_MGR	2086	X'00000826'
MQRC_UNKNOWN_REMOTE_Q_MGR	2087	X'00000827'
MQRC_WAIT_INTERVAL_ERROR	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERROR,	2092	X'0000082C'
MQRC_NOT_OPEN_FOR_PASS_ALL	2093	X'0000082D'
MQRC_NOT_OPEN_FOR_PASS_IDENT	2094	X'0000082E'
MQRC_NOT_OPEN_FOR_SET_ALL	2095	X'0000082F'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_NOT_OPEN_FOR_SET_IDENT	2096	X'00000830'
MQRC_CONTEXT_HANDLE_ERROR	2097	X'00000831'
MQRC_CONTEXT_NOT_AVAILABLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJECT_ALREADY_EXISTS	2100	X'00000834'
MQRC_OBJECT_USZKODZONA	2101	X'00000835'
Problem 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_ANULOWANA	2107	X'0000083B'
BŁĄD MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
BŁĄD FORMAT_MQRC_FORMAT_ERROR	2110	X'0000083E'
MQRC_SOURCE_CCSID_ERROR, BŁĄD	2111	X'0000083F'
MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR, BŁĄD	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, BŁĄD	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_OBCIĘTA	2120	X'00000848'
MQRC_NO_EXTERNAL_UCZESTNICY	2121	X'00000849'
MQRC_W_IPANT_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_NIEDOBÓR	2127	X'0000084F'
MQRC_UOW_IN_PROGRESS	2128	X'00000850'
MQRC_ADAPTER_CONN_LOAD_ERROR	2129	X'00000851'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
Błąd MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
Błąd MQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
BŁĄD MQRC_BO_ERROR	2134	X'00000856'
BŁĄD MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_POWODY	2136	X'00000858'
MQRC_OPEN_FAILED	2137	X'00000859'
MQRC_ADAPTER_DISC_LOAD_ERROR	2138	X'0000085A'
BŁĄD MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED (nie powiodło się)	2140	X'0000085C'
BŁĄD MQRC_DLH_ERROR	2141	X'0000085D'
BŁĄD 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'
BŁĄD MQRC_IIH_ERROR	2148	X'00000864'
BŁĄD MQRC_PCF_ERROR	2149	X'00000865'
BŁĄD MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
BŁĄD 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_ZATRZYMYWANIE	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
BŁĄD MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
BŁĄD MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_DUŻE	2190	X'0000088E'
BŁĄD MQRC_TMC_ERROR	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
BŁĄD MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE	2194	X'00000892'
Błąd MQRC_UNEXPECTED_ERROR	2195	X'00000893'
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
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_ZATRZYMYWANIE	2203	X'0000089B'
MQRC_ADAPTER_NOT_AVAILABLE	2204	X'0000089C'
BŁĄD MQRC_MSG_ID_ERROR	2206	X'0000089E'
MQRC_CORREL_ID_ERROR (BŁĄD)	2207	X'0000089F'
MQRC_FILE_SYSTEM_ERROR	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
Błąd MQRC_SOAP_DOTNET_ERROR	2210	X'000008A2'
BŁĄD MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
MQRC_SOAP_URL_ERROR, BŁĄD	2212	X'000008A4'
MQRC_FILE_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHANNEL	2218	X'000008AA'
MQRC_CALL_IN_PROGRESS	2219	X'000008AB'
MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_ACTIVE	2222	X'000008AE'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
BŁĄD MQRC_CFH_ERROR	2235	X'000008BB'
MQRC_CFIL_ERROR,	2236	X'000008BC'
BŁĄD MQRC_CFIN_ERROR	2237	X'000008BD'
BŁĄD 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'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
MQRC_INCONSISTENT_UOW	2245	X'000008C5'
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'
BŁĄD 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'
BŁĄD MQRC_OFFSET_ERROR	2251	X'000008CB'
MQRC_ORIGINAL_LENGTH_ERROR	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NOT_AVAILABLE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE	2259	X'000008D3'
BŁĄD MQRC_XQH_ERROR	2260	X'000008D4'
MQRC_SRC_ENV_ERROR	2261	X'000008D5'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Błąd MQRC_SRC_NAME_ERROR	2262	X'000008D6'
MQRC_DEST_ENV_ERROR (BŁĄD)	2263	X'000008D7'
MQRC_DEST_NAME_ERROR	2264	X'000008D8'
BŁĄD MQRC_TM_ERROR	2265	X'000008D9'
MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_INHIBITED	2268	X'000008DC'
MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_USE	2271	X'000008DF'
MQRC_PARTIALLY_PRZEKSZTAŁCONA	2272	X'000008E0'
MQRC_CONNECTION_ERROR, BŁĄD	2273	X'000008E1'
Błąd MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
BŁĄD MQRC_CD_ERROR	2277	X'000008E5'
BŁĄD MQRC_CLIENT_CONN_ERROR	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
BŁĄD MQRC_HCONFIG_ERROR	2280	X'000008E8'
MQRC_FUNCTION_ERROR	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
MQRC_TERMINATION_FAILED	2287	X'000008EF'
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'
MQRC_SERVICE_ERROR,	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
MQRC_USER_ID_NOT_AVAILABLE	2291	X'000008F3'
MQRC_UNKNOWN_ENTITY,	2292	X'000008F4'
MQRC_UNKNOWN_AUTH_ENTITY	2293	X'000008F5'
MQRC_UNKNOWN_REF_OBJECT	2294	X'000008F6'
MQRC_CHANNEL_ACTIVATED	2295	X'000008F7'
MQRC_CHANNEL_NOT_ACTIVATED	2296	X'000008F8'
MQRC_UOW_ANULOWANO	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED	2298	X'000008FA'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
BŁĄD 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'
Funkcja MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENT	2309	X'00000905'
MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_OBCIĘTY	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'
BŁĄD MQRC_HBAG_ERROR	2320	X'00000910'
Brak parametru MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE	2322	X'00000912'
Błąd MQRC_STRING_LENGTH_ERROR	2323	X'00000913'
MQRC_INQUIRY_COMMAND_ERROR	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPPORTED	2325	X'00000915'
MQRC_BAG_WRONG_TYPE	2326	X'00000916'
MQRC_ITEM_TYPE_ERROR	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE,	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
MQRC_CODED_CHAR_SET_ID_BŁĄD	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
BŁĄD MQRC_WIH_ERROR	2333	X'0000091D'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD 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_ZWOLNIONY	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_DLA_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_NIEZGODNY	2360	X'00000938'
MQRC_NEXT_RECORD_ERROR	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_OSIĄGNĘŁA	2362	X'0000093A'
MQRC_MSG_NOT_MATCHED	2363	X'0000093B'
MQRC_JMS_FORMAT_ERROR,	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED	2365	X'0000093D'
MQRC_WRONG_CF_LEVEL	2366	X'0000093E'
MQRC_CONFIG_CREATE_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_NAZWA_IPANT_NOT_DEFINED	2372	X'00000944'
MQRC_CF_STRUC_NIE POWIODŁO SIĘ	2373	X'00000945'
BŁĄD 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'
Błąd MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
Błąd MQRC_CRYPTOHARDWARE_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_ZAINICJOWANY	2391	X'00000957'
BŁĄD MQRC_SSL_CONFIG_ERROR	2392	X'00000958'
MQRC_SSL_INITIALIZATION_ERROR,	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
BŁĄD 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-BŁĄD	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_ODWOŁANE	2401	X'00000961'
MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'
MQRC_UOW_COMMITTED	2408	X'00000968'
BŁĄD 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'
BŁĄD MQRC_CFSF_ERROR	2415	X'0000096F'
MQRC_CFGR_ERROR,	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_GROUP	2417	X'00000971'
Błąd MQRC_FILTER_OPERATOR_ERROR	2418	X'00000972'
MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
BŁĄD MQRC_EPH_ERROR	2420	X'00000974'
Błąd formatu MQRC_RFH_FORMAT_ERROR	2421	X'00000975'
MQRC_CFBF_ERROR	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICT	2423	X'00000977'
Błąd MQRC_SD_ERROR	2424	X'00000978'
Błąd MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
BŁĄD MQRC_STS_ERROR	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE	2429	X'0000097D'
Błąd MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_ERROR	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
Niezgodność MQRC_IDENTITY_MISMATCH	2434	X'00000982'
MQRC_ALTER_SUB_ERROR,	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG,	2437	X'00000985'
BŁĄD MQRC_SRO_ERROR	2438	X'00000986'
MQRC_SUB_NAME_ERROR-BŁĄD	2440	X'00000988'
MQRC_OBJECT_STRING_ERROR,	2441	X'00000989'
Błąd MQRC_PROPERTY_NAME_ERROR	2442	X'0000098A'
MQRC_SEGMENTATION_NOT_ALLOWED	2443	X'0000098B'
MQRC_CBD_ERROR	2444	X'0000098C'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRC_CTLO_ERROR	2445	X'0000098D'
MQRC_NO_CALLBACKS_ACTIVE	2446	X'0000098E'
MQRC_CALLBACK_NOT_ZAREJESTROWANY	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'
MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
BŁĄD MQRC_HMSG_ERROR	2460	X'0000099C'
BŁĄD MQRC_CMHO_ERROR	2461	X'0000099D'
BŁĄD MQRC_DMHO_ERROR	2462	X'0000099E'
BŁĄD MQRC_SMPO_ERROR	2463	X'0000099F'
BŁĄD MQRC_IMPO_ERROR	2464	X'000009A0'
MQRC_PROPERTY_NAME_TOO_BIG	2465	X'000009A1'
MQRC_PROP_VALUE_NOT_CONVERTED	2466	X'000009A2'
MQRC_PROP_TYPE_NOT_SUPPORTED	2467	X'000009A3'
MQRC_PROPERTY_VALUE_TOO_BIG	2469	X'000009A5'
MQRC_PROP_CONV_NOT_SUPPORTED	2470	X'000009A6'
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_ZACHOWANE	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CHANGED	2480	X'000009B0'
BŁĄD MQRC_DMPO_ERROR	2481	X'000009B1'
BŁĄD MQRC_PD_ERROR	2482	X'000009B2'
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'
BŁĄD MQRC_OPERATION_ERROR	2488	X'000009B8'
BŁĄD MQRC_BMHO_ERROR	2489	X'000009B9'
MQRC_UNSUPPORTED_PROPERTY,	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERTED	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'
MQRC_MODULE_NOT_FOUND	2495	X'000009BF'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Niepoprawna wartość MQRC_MODULE_INVALID	2496	X'000009C0'
MQRC_MODULE_ENTRY_NOT_FOUND	2497	X'000009C1'
MQRC_MIXED_CONTENT_NOT_ALLOWED	2498	X'000009C2'
MQRC_MSG_HANDLE_IN_USE	2499	X'000009C3'
MQRC_HCONN_ASYNC_ACTIVE	2500	X'000009C4'
MQRC_MHBO_ERROR	2501	X'000009C5'
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_INHIBITED	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'
BŁĄD 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'
Błąd 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_DOSTARZONEGO	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_STOPPED	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBITED	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERATION_NOT_ALLOWED	2534	X'000009E6'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD 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_PUBLIKACJA	2541	X'000009ED'
MQRC_ALREADY_DOŁĄCZYŁ (dołączył)	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 (BŁĄD)	2592	X'00000A20'
Błąd MQRC_PASSWORD_PROTECTION_ERROR	2594	X'00000A22'
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'
Błąd MQRC_ENCODING_ERROR	6106	X'000017DA'
MQRC_STRUC_ID_BŁĄD	6107	X'000017DB'
MQRC_NULL_POINTER	6108	X'000017DC'
MQRC_NO_CONNECTION_REFERENCE	6109	X'000017DD'
MQRC_NO_BUFFER	6110	X'000017DE'
MQRC_BINARY_DATA_LENGTH_ERROR	6111	X'000017DF'
MQRC_BUFFER_NOT_AUTOMATIC	6112	X'000017E0'
MQRC_INSUFFICIENT_BUFFER	6113	X'000017E1'
MQRC_INSUFFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_OBCIĘTY	6115	X'000017E3'
MQRC_ZERO_LENGTH	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
MQRC_NEGATIVE_OFFSET	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
MQRC_CONTEXT_OPEN_ERROR	6122	X'000017EA'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_STRUC_LENGTH_ERROR	6123	X'000017EB'
MQRC_NOT_CONNECTED	6124	X'000017EC'
MQRC_NOT_OPEN	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
MQRC_INCONSISTENT_OPEN_OPTIONS	6127	X'000017EF'
MQRC_WRONG_VERSION	6128	X'000017F0'
MQRC_REFERENCE_ERROR	6129	X'000017F1'

MQRCCF_* (kody przyczyny nagłówka formatu komendy)

Więcej informacji na temat odpowiedzi programisty zawiera sekcja [Kody przyczyny PCF](#).

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_CFH_TYPE_ERROR-BŁĄD	3001	X'00000BB9'
MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
MQRCCF_CFH_VERSION_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_SEQ_NUMBER_ERR	3004	X'00000BBC'
MQRCCF_CFH_CONTROL_ERROR	3005	X'00000BBD'
MQRCCF_CFH_PARM_COUNT_ERROR	3006	X'00000BBE'
MQRCCF_CFH_COMMAND_ERROR	3007	X'00000BBF'
MQRCCF_COMMAND_NIE POWIODŁO SIĘ	3008	X'00000BC0'
MQRCCF_CFIN_LENGTH_BŁĄD	3009	X'00000BC1'
MQRCCF_CFST_LENGTH_ERROR-BŁĄD	3010	X'00000BC2'
MQRCCF_CFST_STRING_LENGTH_ERR	3011	X'00000BC3'
MQRCCF_FORCE_VALUE_ERROR-BŁĄD	3012	X'00000BC4'
MQRCCF_STRUCTURE_TYPE_BŁĄD	3013	X'00000BC5'
MQRCCF_CFIN_PARM_ID_BŁĄD-BŁĄD	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_BŁĄD-BŁĄD	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-BŁĄD	3022	X'00000BCE'
MQRCCF_MD_FORMAT_ERROR	3023	X'00000BCF'
MQRCCF_CFSL_LENGTH_ERROR-BŁĄD	3024	X'00000BD0'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_REPLACE_VALUE_ERROR-BŁĄD	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-BŁĄD	3029	X'00000BD5'
MQRCCF_MODE_VALUE_ERROR-BŁĄD	3029	X'00000BD5'
MQRCCF_MSG_SEQ_NUMBER_ERROR	3030	X'00000BD6'
MQRCCF_PING_DATA_COUNT_ERROR (BŁĄD)	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_PARM_ID_ERROR	3033	X'00000BD9'
MQRCCF_CHANNEL_TYPE_ERROR-BŁĄD	3034	X'00000BDA'
MQRCCF_PARM_SEQUENCE_ERROR	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
Błąd MQRCCF_BATCH_SIZE_ERROR	3037	X'00000BDD'
MQRCCF_DISC_INT_ERROR	3038	X'00000BDE'
MQRCCF_SHORT_RETRY_ERROR-BŁĄD	3039	X'00000BDF'
MQRCCF_SHORT_TIMER_ERROR-BŁĄD	3040	X'00000BE0'
BŁĄD MQRCCF_LONG_RETRY_ERROR	3041	X'00000BE1'
Błąd MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'
MQRCCF_SEQ_NUMBER_WRAP_ERROR	3043	X'00000BE3'
MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'
MQRCCF_PUT_AUTH_ERROR	3045	X'00000BE5'
MQRCCF_PURGE_VALUE_ERROR-BŁĄD	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_BŁĄD	3047	X'00000BE7'
MQRCCF_MSG_OBCIĘTO	3048	X'00000BE8'
MQRCCF_CCSID_ERROR-BŁĄD	3049	X'00000BE9'
MQRCCF_ENCODING_ERROR-BŁĄD	3050	X'00000BEA'
MQRCCF_QUEUES_VALUE_ERROR (BŁĄD)	3051	X'00000BEB'
MQRCCF_DATA_CONV_VALUE_ERROR	3052	X'00000BEC'
MQRCCF_INDOUBT_VALUE_ERROR (BŁĄD)	3053	X'00000BED'
MQRCCF_ESCAPE_TYPE_ERROR-BŁĄD	3054	X'00000BEE'
MQRCCF_REPOS_VALUE_ERROR-BŁĄD	3055	X'00000BEF'
MQRCCF_CHANNEL_TABLE_ERROR	3062	X'00000BF6'
MQRCCF_MCA_TYPE_ERROR-BŁĄD	3063	X'00000BF7'
MQRCCF_CHL_INST_TYPE_ERROR	3064	X'00000BF8'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_CHL_STATUS_NOT_FOUND	3065	X'00000BF9'
MQRCCF_CFSL_DUPLICATE_PARM	3066	X'00000BFA'
MQRCCF_CFSL_TOTAL_LENGTH_ERROR-BŁĄD	3067	X'00000BFB'
MQRCCF_CFSL_COUNT_ERROR (BŁĄD)	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_DELETED	3070	X'00000BFE'
MQRCCF_STREAM_ERROR	3071	X'00000BFF'
BŁĄD MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NOT_ZAREJESTROWANY	3073	X'00000C01'
Błąd MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_INCORRECT_STREAM	3075	X'00000C03'
MQRCCF_Q_NAME_ERROR-BŁĄD	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-BŁĄD	3080	X'00000C08'
MQRCCF_NOT_AUTHORIZED (autoryzowany)	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'
MQRCCF_REG_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR,	3084	X'00000C0C'
MQRCCF_UNKNOWN_BROKER	3085	X'00000C0D'
MQRCCF_Q_MGR_CCSID_ERROR-BŁĄD	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR,	3087	X'00000C0F'
MQRCCF_CLUSTER_NAME_CONFLICT	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ACTION_VALUE_ERROR-BŁĄD	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_NETBIOS_NAME_ERROR-BŁĄD	3093	X'00000C15'
Komenda 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 (BŁĄD)	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_WRONG_USER	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
MQRCCF_SUB_IDENTITY_ERROR-BŁĄD	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED	3156	X'00000C54'
MQRCCF_ALREADY_DOŁĄCZYŁ (a)	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-BŁĄD	3163	X'00000C5B'
MQRCCF_ALLOC_RETRY_BŁĄD	3164	X'00000C5C'
MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
MQRCCF_NUMER_PORTU_BŁĘDU	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ACTIVE	3168	X'00000C60'
BRAK DANYCH MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
MQRCCF_PROFILE_NAME_ERROR	3170	X'00000C62'
MQRCCF_AUTH_VALUE_ERROR-BŁĄD	3171	X'00000C63'
Brak elementu MQRCCF_AUTH_VALUE_MISSING	3172	X'00000C64'
MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
MQRCCF_CONNECTION_ID_ERROR-BŁĄD	3174	X'00000C66'
MQRCCF_LOG_TYPE_ERROR-BŁĄD	3175	X'00000C67'
MQRCCF_PROGRAM_NOT_AVAILABLE	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_FAILED	3177	X'00000C69'
MQRCCF_NONE_FOUND	3200	X'00000C80'
MQRCCF_SECURITY_SWITCH_OFF	3201	X'00000C81'
MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'
MQRCCF_PARM_CONFLICT	3203	X'00000C83'
MQRCCF_COMMAND_INHIBITED	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_OBJECT_NAME_RESTRICTED	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_EXCEEDED	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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 (BŁĄD)	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'
Błąd MQRCCF_COMMAND_SCOPE_ERROR	3225	X'00000C99'
MQRCCF_COMMAND_REPLY_ERROR-BŁĄD	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED	3227	X'00000C9B'
BRĄK 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 (ROZPOCZĘTE)	3233	X'00000CA1'
MQRCCF_LISTENER_STOPPED	3234	X'00000CA2'
MQRCCF_CHANNEL_ERROR-BŁĄD	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR-BŁĄD	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_PARM_ID_ERROR	3240	X'00000CA8'
MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
MQRCCF_CFIF_OPERATOR_ERROR	3242	X'00000CAA'
MQRCCF_CFIF_PARM_ID_BŁĄD	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_PARM_ID_BŁĄD	3247	X'00000CAF'
MQRCCF_TOO_MANY_FILTERS	3248	X'00000CB0'
MQRCCF_LISTENER_RUNNING	3249	X'00000CB1'
MQRCCF_LSTR_STATUS_NOT_FOUND	3250	X'00000CB2'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_SERVICE_RUNNING	3251	X'00000CB3'
MQRCCF_SERV_STATUS_NOT_FOUND	3252	X'00000CB4'
MQRCCF_SERVICE_STOPPED	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
MQRCCF_CFBS_LENGTH_ERROR	3255	X'00000CB7'
MQRCCF_CFBS_PARM_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_BŁĄD-BŁĄD	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'
BŁĄD 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-BŁĄD	3315	X'00000CF3'
MQRCCF_DEST_NAME_ERROR-BŁĄD	3316	X'00000CF4'
MQRCCF_INVALID_DESTINATION	3317	X'00000CF5'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_PUBSUB_INHIBITED	3318	X'00000CF6'
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_WRONG_CHLAUTH_TYPE	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_WRONG_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_WRONG_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_WRONG_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICT	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_EXCEEDED	3344	X'00000D10'
BŁĄD 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-BŁĄD	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR (BŁĄD)	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_INVALID_PROTOCOL	3365	X'00000D25'
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-BŁĄD	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-BŁĄD	4011	X'00000FAB'
MQRCCF_CONNECTION_REFUSED	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR (BŁĄD)	4013	X'00000FAD'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_SEND_NIE POWIODŁO SIĘ	4014	X'00000FAE'
MQRCCF_RECEIVED_DATA_ERROR	4015	X'00000FAF'
MQRCCF_RECEIVE_NIE POWIODŁO SIĘ	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'
MQRCCF_NO_STORAGE	4018	X'00000FB2'
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED	4020	X'00000FB4'
MQRCCF_BIND_NIE POWIODŁO SIĘ	4024	X'00000FB8'
MQRCCF_CHANNEL_INDOUBT	4025	X'00000FB9'
Komenda 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-BŁĄD	4030	X'00000FBE'
MQRCCF_CHANNEL_IN_USE	4031	X'00000FBF'
MQRCCF_CHANNEL_NOT_FOUND	4032	X'00000FC0'
MQRCCF_UNKNOWN_REMOTE_CHANNEL	4033	X'00000FC1'
MQRCCF_REMOTE_QM_UNAVAILABLE	4034	X'00000FC2'
MQRCCF_REMOTE_QM_TERMINATING	4035	X'00000FC3'
MQRCCF_MQINQ_FAILED	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NOT_AVAILABLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_WRONG_CHANNEL_TYPE	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
MQRCCF_CHANNEL_NAME_ERROR-BŁĄD	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
MQRCCF_MCA_NAME_ERROR-BŁĄD	4047	X'00000FCF'
MQRCCF_SEND_EXIT_NAME_ERROR-BŁĄD	4048	X'00000FD0'
MQRCCF_SEC_EXIT_NAME_ERROR-BŁĄD	4049	X'00000FD1'
MQRCCF_MSG_EXIT_NAME_ERROR-BŁĄD	4050	X'00000FD2'
MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_WRONG_TYPE	4052	X'00000FD4'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
BŁĄD MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_NIE POWIODŁO SIĘ	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ACTIVE	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
Błąd MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NOT_AVAILABLE	4068	X'00000FE4'
MQRCCF_MR_COUNT_ERROR (BŁĄD)	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-BŁĄD	4075	X'00000FEB'
MQRCCF_NPM_SPEED_WRONG_TYPE	4076	X'00000FEC'
MQRCCF_HB_INTERVAL_ERROR	4077	X'00000FED'
MQRCCF_HB_INTERVAL_WRONG_TYPE	4078	X'00000FEE'
BŁĄD MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_WRONG_TYPE	4080	X'00000FF0'
MQRCCF_CHAD_EVENT_ERROR-BŁĄD	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 (BŁĄD)	4086	X'00000FF6'
MQRCCF_BATCH_INT_WRONG_TYPE	4087	X'00000FF7'
Błąd MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_NET_PRIORITY_WRONG_TYPE	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_PEER_NAME_ERROR-BŁĄD	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED	4095	X'00000FFF'

MQRCN_* (stałe połączenia klienta)

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

MQRCVTIME_* (typy limitu czasu odbierania)

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

MQREADA_* (odczytywanie wartości nagłówka)

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

MQRECORDING_* (opcje rejestrowania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRECORDING_DISABLED	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_* (opcje rejestracji publikowania/subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQREGO_NONE	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY,	1	X'00000001'
MQREGO_ANONYMOUS	2	X'00000002'
MQREGO_LOCAL	4	X'00000004'
MQREGO_DIRECT_REQUESTS,	8	X'00000008'
MQREGO_NEW_PUBLICATIONS_ONLY	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY	32	X'00000020'
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_STREAM_NAME	128	X'00000080'
MQREGO_INFORM_IF_ZACHOWANY	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'
MQREGO_NON_PERSISTENT	1024	X'00000400'
MQREGO_PERSISTENT	2048	X'00000800'
MQREGO_PERSISTENT_AS_PUBLISH	4096	X'00001000'
MQREGO_PERSISTENT_AS_Q	8192	X'00002000'
MQREGO_NAZWA_ADD_NAME	16384	X'00004000'
MQREGO_NO_ALTERATION	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY	524288	X'00080000'
MQREGO_VARIABLE_USER_ID,	1048576	X'00100000'
MQREGO_LOCKED	2097152	X'00200000'

MQRFH_* (reguły i struktura nagłówka formatowania i flagi)

Reguły i struktura nagłówka formatowania

Nazwa	Struktura
MQRFH_STRUC_ID	"RFH↵"
MQRFH_STRUC_ID_ARRAY	'R','F','H','↵'

Uwaga: Symbol ↵ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRFH_VERSION_1	1	X'00000001'
MQRFH_VERSION_2	2	X'00000002'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRFH_STRUC_LENGTH_FIXED	32	X'00000020'
MQRFH_STRUC_LENGTH_FIXED_2	36	X'00000024'

Flagi reguł i formatowania nagłówka

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

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

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (nazwy znaczników znaczników opcji publikowania/subskrypcji)

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

MQRFH2_* (nazwy znaczników XML znaczników opcji publikowania/subskrypcji)

MQRFH2_PUBSUB_CMD_FOLDER_B	"<psc>"
MQRFH2_PUBSUB_CMD_FOLDER_E	"</psc>"
MQRFH2_PUBSUB_RESP_FOLDER_B	"<pscr>"
MQRFH2_PUBSUB_RESP_FOLDER_E	"</pscr>"
MQRFH2_MSG_CONTENT_FOLDER_B	"<mcd>"
MQRFH2_MSG_CONTENT_FOLDER_E	"</mcd>"
MQRFH2_USER_FOLDER_B	"<usr>"
MQRFH2_USER_FOLDER_E	"</usr>"

MQRL_* (Zwrócony Długość)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRL_NIEZDEFINIOWANY	-1	X'FFFFFFFF'

MQRMH_* (struktura nagłówka komunikatu odniesienia)

Nazwa	Struktura
MQRMH_STRUC_ID	"RMH"
MQRMH_STRUC_ID_ARRAY	'R','M','H',' '

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

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

MQRMHF_ * (Flagi nagłówka komunikatu odwołania)

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

MQRO_ * (opcje raportu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRO_EXCEPTION	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
MQRO_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 (Identyfikator 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_WAŻNOŚCI	16384	X'00004000'
MQRO_NONE	0	X'00000000'

MQRO_* (Maski opcji raportu)

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

MQROUTE_* (trasa śledzenia)**Maksymalna liczba działań śledzenia trasy śledzenia (MQIACF_MAX_ACTIVITIES)**

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Szczegóły trasy śledzenia (MQIACF_ROUTE_DETAIL)

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

Przekazywanie trasy śledzenia (MQIACF_ROUTE_FORWARDING)

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

Dostarczanie trasy śledzenia (MQIACF_ROUTE_DELIVERY)

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

Kumulacja trasy śledzenia (MQIACF_ROUTE_AKUMULACJA)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_ACCUMULATE_NONE (brak)	65539	X'00010003'
Komunikat MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_AND_REPLY	65541	X'00010005'

MQRP_* (Opcje zastępowania formatu komend)

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

MQRQ_* (Kwalifikatory przyczyny formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRQ_CONN_NOT_AUTHORIZED (autoryzowany)	1	X'00000001'
MQRQ_OPEN_NOT_AUTHORIZED (autoryzowany)	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED (AUTORYZOWANY)	4	X'00000004'
MQRQ_Q_MGR_ZATRZYMYWANIE	5	X'00000005'
MQRQ_Q_MGR_QUIESCING,	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_BRIDGE_STOPPED_OK	11	X'0000000B'
Błąd MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
Błąd MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR,	15	X'0000000F'
BŁĄD MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED (autoryzowany)	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED (autoryzowany)	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'
MQRQ_SYS_CONN_NOT_AUTHORIZED	20	X'00000014'
MQRQ_CHANNEL_BLOCKED_ADDRESS	21	X'00000015'
MQRQ_CHANNEL_BLOCKED_USERID	22	X'00000016'
MQRQ_CHANNEL_BLOCKED_NOACCESS	23	X'00000017'
MQRQ_MAX_ACTIVE_CHANNELS	24	X'00000018'
MQRQ_MAX_CHANNELS	25	X'00000019'
MQRQ_SVRCONN_INST_LIMIT	26	X'0000001A'
MQRQ_CLIENT_INST_LIMIT!	27	X'0000001B'
MQRQ_CAF_NOT_INSTALLED	28	X'0000001C'

MQRT_* (typy odświeżania formatu komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
KONFIGURACJA MQRT_CONFIGURATION	1	X'00000001'
MQRT_TERMIN	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_* (tylko żądanie)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_* (uwierzytelnianie klienta SSL)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCA_REQUIRED	0	X'00000000'
MQSCA_OPTIONAL	1	X'00000001'

MQSCO_* (opcje konfiguracji SSL)

Struktura opcji konfiguracji protokołu SSL

Nazwa	Struktura
Identyfikator MQSCO_STRUC_ID	"SC0~"
Tabela MQSCO_STRUC_ID_ARRAY	'S','C','O','~'

Uwaga: Symbol ~ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
MQSCO_CURRENT_VERSION	4	X'00000004'

Uwaga: Symbol ~ reprezentuje pojedynczy pusty znak.

Liczba resetowań klucza opcji konfiguracji SSL

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Zasięg definicji kolejki formatu komend

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

MQSCOPE_* (zasięg publikowania)

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

MQSCYC_* (przypadek zabezpieczeń)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCYC_GÓRNY	0	X'00000000'
MQSCYC_MIESZANY	1	X'00000001'

MQSD_* (struktura deskryptora obiektu)

Nazwa	Struktura
MQSD_STRUC_ID	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

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

MQSECITEM_* (elementy zabezpieczeń w formacie komend)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSECITEM_ALL	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMLS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECSW_* (Przełączniki bezpieczeństwa w formacie komend i stany przełączników)

Przełączniki bezpieczeństwa w formacie komend

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSECSW_PROCESS,	1	X'00000001'
MQSECSW_NAMELIST,	2	X'00000002'
Kolejka 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'
PODSYSTEM MQSECSW_SUBSYSTEM	10	X'0000000A'
MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Stany przełączników zabezpieczeń formatu komend

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSECSW_OFF_FOUND	21	X'00000015'
MQSECSW_ON_FOUND	22	X'00000016'
MQSECSW_OFF_NOT_FOUND	23	X'00000017'
MQSECSW_ON_NOT_FOUND	24	X'00000018'
BŁĄD MQSECSW_OFF_ERROR	25	X'00000019'
MQSECSW_ON_PRZESŁONIĘTE	26	X'0000001A'

MQSECTYPE_* (typy zabezpieczeń formatu komendy)

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

MQSEG_* (Segmentacja)

Nazwa	Wartość
MQSEG_INHIBITED	' - '
MQSEG_ALLOWED	' A '

Uwaga: Symbol - reprezentuje pojedynczy pusty znak.

MQSEL_* (Specjalne wartości selektorów)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (typy Selektorów)

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

MQSID_* (identyfikator zabezpieczeń)

Nazwa	Wartość
MQSID_NONE	X'00...00' (40 zer)
MQSID_NONE_ARRAY	'\0', '\0', ... (40 zer)

MQSIDT_* (typy identyfikatorów zabezpieczeń)

Nazwa	Wartość szesnastkowa
MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_* (Ustaw opcje i strukture właściwości komunikatu)

Ustaw strukture opcji właściwości komunikatu

Nazwa	Struktura
MQSMPO_STRUC_ID,	"SMPO"
Tablica MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_CURRENT_VERSION	1	X'00000001'

Ustaw opcje właściwości komunikatu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSMPO_SET_FIRST	0	X'00000000'
MQSMPO_SET_PROP_UNDER_CURSOR	1	X'00000001'
MQSMPO_SET_PROP_AFTER_CURSOR	2	X'00000002'
MQSMPO_APPEND_PROPERTY,	4	X'00000004'
MQSMPO_SET_PROP_BEFORE_CURSOR	8	X'00000008'
MQSMPO_NONE,	0	X'00000000'

MQSO_* (opcje subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSO_BRAK	0	X'00000000'
MQSO_NON_DURABLE,	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQSO_ALTER	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE,	8	X'00000008'
MQSO_GROUP_SUB	16	X'00000010'
MQSO_MANAGED	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
ŻĄDANIE MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLICATIONS_ONLY (TYLKO)	4096	X'00001000'
MQSO_FAIL_IF QUIESCING	8192	X'00002000'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (Dostępność punktu synchronizacji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

MQSQQM_* (nazwa menedżera kolejek współużytkowanych kolejek)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSQQM_USE	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_* (działanie)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSR_ACTION_PUBLICATION	1	X'00000001'

MQSRO_* (struktura opcji żądania subskrypcji)

Nazwa	Struktura
MQSRO_STRUC_ID,	"SR0¬"
MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (status segmentu)

Nazwa	Struktura
MQSS_NOT_A_SEGMENT	'¬'

Nazwa	Struktura
Segment MQSS_SEGMENT	'S'
MQSS_LAST_SEGMENT,	'L'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

MQSSL_* (wymagania SSL FIPS)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_* (opcje Stat)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSTAT_TYPE_ASYNC_ERROR,	0	X'00000000'
MQSTAT_TYPE_RECONNECTION	0	X'00000000'
MQSTAT_TYPE_RECONNECTION_ERROR	0	X'00000000'

MQSTS_* (struktura struktury raportowania statusu)

Nazwa	Struktura
MQSTS_STRUC_ID	"STAT"
MQSTS_STRUC_ID_ARRAY	'S','T','A','T'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSTS_VERSION_1	1	X'00000001'
MQSTS_CURRENT_VERSION	1	X'00000001'

MQSUB_* (subskrypcje trwałe)

Subskrypcje stałe

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBITED	2	X'00000002'

Subskrypcje stałe

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_* (typy subskrypcji formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Funkcja API MQSUBTYPE_API	1	X'00000001'
Administrator MQSUBTYPE_ADMIN	2	X'00000002'
Proxy MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
UŻYTKOWNIK MQSUBTYPE_USER	-2	X'FFFFFFFE'

MQSUS_* (Status zawieszenia formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSUS_YES	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_* (usługa)

Typy usług

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSVC_TYPE_COMMAND	0	X'00000000'
MQSVC_TYPE_SERVER	1	X'00000001'

Elementy sterujące usługi

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

Status usługi

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSVC_STATUS_STOPPED	0	X'00000000'
MQSVC_STATUS_URUCHAMIANIE	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
MQSVC_STATUS_ZATRZYMYWANIE	3	X'00000003'
MQSVC_STATUS_RETRYING	4	X'00000004'

MQSYNCPPOINT_* (Format komendy-wartości punktu synchronizacji dla migracji publikowania/subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSYNCPPOINT_YES	0	X'00000000'
IFSYNCPPOINT_IFPER	1	X'00000001'

MQSYSP_* (wartości parametrów systemowych formatu komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSYSP_NO	0	X'00000000'
MQSYSP_TAK	1	X'00000001'
MQSYSP_EXTENDED	2	X'00000002'
MQSYSP_TYPE_INITIAL	10	X'0000000A'
MQSYSP_TYPE_SET	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY,	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS	13	X'0000000D'
MQSYSP_TYPE_ARCHIVE_TAPE	14	X'0000000E'
MQSYSP_ALLOC_BLK	20	X'00000014'
MQSYSP_ALLOC_TRK	21	X'00000015'
MQSYSP_ALLOC_CYL	22	X'00000016'
MQSYSP_STATUS_BUSY	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_AVAILABLE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
MQSYSP_STATUS_COPYING_LOG,	36	X'00000024'

MQTA_* (atrybuty tematu)

Znaki wieloznaczne

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BLOKADA MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

Subskrypcje dozwolone

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTA_SUB_AS_PARENT,	0	X'00000000'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTA_SUB_INHIBITED	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

Propagacja podrzędna proxy

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Dozwolone publikacje

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTA_PUB_AS_PARENT,	0	X'00000000'
MQTA_PUB_INHIBITED	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_* (Elementy sterujące wyzwalacza)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_* (Keepalive TCP)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_* (typy stosu TCP)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_* (jednostki czasu w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_* (Struktura komunikatu wyzwalacza)

Nazwa	Struktura
MQTM_STRUC_ID	"TM--"

Nazwa	Struktura
MQTM_STRUC_ID_ARRAY	'T','M',' ',' '

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

MQTMC_* (Struktura formatu znaków komunikatu wyzwalacza)

Nazwa	Struktura
MQTMC_STRUC_ID	"TMC"
MQTMC_STRUC_ID_ARRAY,	'T','M','C',' '
MQTMC_VERSION_1	"1"
MQTMC_VERSION_2	"2"
MQTMC_CURRENT_VERSION	"2"
MQTMC_VERSION_1_ARRAY	' ',' ',' ','1'
MQTMC_VERSION_2_ARRAY	' ',' ',' ','2'
MQTMC_CURRENT_VERSION_ARRAY	' ',' ',' ','2'

MQTOPT_* (typ tematu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTOPT_LOCAL	0	X'00000000'
Klaster MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_* (Autostart śledzenia inicjatora kanału)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_YES	1	X'00000001'

MQTScope_* (zasięg subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTScope_QMGR	1	X'00000001'
MQTScope_ALL	2	X'00000002'

MQTT_* (typy wyzwalaczy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTT_NONE	0	X'00000000'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_ * (typy danych właściwości)

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

MQUA_ * (selektory atrybutów użytkowników publikowania/subskrypcji)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUA_FIRST	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_ * (Obsługa identyfikatora użytkownika w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_YES	1	X'00000001'

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

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_* (Stany UOW w formacie komendy)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUOWST_BRAK	0	X'00000000'
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARED	2	X'00000002'
MQUOWST_UNRESOLVED	3	X'00000003'

MQUOWT_* (Typy UOW w formacie komendy)

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

MQUS_* (Usages kolejki)

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

MQUSAGE_* (format strony-wartości użycia zestawu stron i wartości użycia zestawu danych)

Format komendy-wartości użycia zestawu stron

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUSAGE_PS_AVAILABLE	0	X'00000000'
MQUSAGE_PS_DEFINED	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_PS_SUSPENDED	4	X'00000004'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SYSTEM	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Wartości użycia zestawu danych w formacie komendy

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUSAGE_DS_OLDEST_ACTIVE_UOW	10	X'0000000A'
MQUSAGE_DS_OLDEST_PS_RECOVERY	11	X'0000000B'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUSAGE_DS_OLDEST_CF_RECOVERY	12	X'0000000C'

MQVL_* (długość wartości)

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

MQVU_* (zmienna ID użytkownika)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQVU_FIXED_USER,	1	X'00000001'
MQVU_ANY_USER,	2	X'00000002'

MQWDR_* (struktura rekordu miejsca docelowego wyjścia obciążenia klastra)

Nazwa	Struktura
ID_STRUC_na_potrzeby	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W', 'D', 'R', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
BIEŻĄCA_BIEŻĄCA_WERSJA	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
BIEŻĄCA_DŁUGOŚĆ_PRACY	136	X'00000088'

MQWI_* (odstęp czasu oczekiwania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWI_UNLIMITED	-1	X'FFFFFFFF'

MQWIH_* (struktura nagłówka informacji obciążenia i flagi)

Struktura nagłówka informacji o obciążeniu

Nazwa	Struktura
MQWIH_STRUC_ID	"WIH¬"
Tablica MQWIH_STRUC_ID_ARRAY	'W', 'I', 'H', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

Flagi nagłówka informacji o obciążeniu

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWIH_NONE	0	X'00000000'

MQWQR_* (Struktura rekordu kolejki wyjścia obciążenia klastra)

Nazwa	Struktura
ID_STRU_aplikacji MQWQR_STRUC	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W', 'Q', 'R', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
MQWQR_CURRENT_VERSION	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_CURRENT_LENGTH	212	X'000000D4'

MQWS_* (schemat wieloznaczny)

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

MQWXP_* (struktura parametru wyjścia obciążenia klastra)

MQWXP_* (struktura parametru wyjścia obciążenia klastra)

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

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
MQWXP_CURRENT_VERSION	4	X'00000004'

MQWXP_ * (Flagi obciążenia klastra)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Komponent MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Odsyłacze pokrewne

“Pola w produkcie MQWXP -struktura parametru wyjścia obciążenia klastra” na stronie 1634
Opis pól w strukturze parametru wyjścia obciążenia klastra MQWXP

MQXACT_ * (typy Caller API)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_ * (komendy obsługi wyjścia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (Wyjdź z odpowiedzi)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXCC_OK	0	X'00000000'
MQXCC_SUPPRESS_FUNCTION	-1	X'FFFFFFFF'
MQXCC_SKIP_FUNCTION,	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
Niepowodzenie MQXCC_FAILED	-8	X'FFFFFFF8'

MQXDR_* (wyjście z odpowiedzi)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_* (środowiska)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXE_INNY	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_* (Zarejestruj strukturę opcji punktu wejścia i opcje wyjścia)

Zarejestruj strukturę opcji punktu wejścia

Nazwa	Struktura
MQXEPO_STRUC_ID	"XEPO"
Tablica MQXEPO_STRUC_ID_ARRAY	'X', 'E', 'P', 'O'

Uwaga: Symbol – reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_CURRENT_VERSION	1	X'00000001'

Opcje wyjścia

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXEPO_NONE	0	X'00000000'

MQXF_ * (Identyfikatory Funkcji API)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
Komenda MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXP_* (struktura parametru wyjścia przekraczania interfejsu API)

Nazwa	Struktura
Identyfikator MQXP_STRUC_ID	"XP¬"
MQXP_STRUC_ID_ARRAY	'X','P','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXP_VERSION_1	1	X'00000001'

MQXPDA_* (obszar określania problemu)

Nazwa	Wartość
MQXPDA_NONE	X'00...00' (48 zer)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 zer)

MQXPT_* (typy transportu)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXPT_ALL	-1	X'FFFFFFFF'
MQXPT_LOCAL	0	X'00000000'
MQXPT_LU62	1	X'00000001'
TCP MQXPT_TCP	2	X'00000002'
MQXPT_NETBIOS	3	X'00000003'
MQXPT_SPX	4	X'00000004'
MQXPT_DECNET	5	X'00000005'
MQXPT_UDP	6	X'00000006'

MQXQH_* (Struktura nagłówka kolejki transmisji)

Nazwa	Struktura
MQXQH_STRUC_ID	"XQH¬"
MQXQH_STRUC_ID_ARRAY,	'X','Q','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_* (Przyczyny Wyjścia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXR_PRZED	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'
MQXR_INIT	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
MQXR_XMIT	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INIT_SEC	16	X'00000010'
MQXR_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
MQXR_ACK_RECEIVED	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'
MQXR_SEC_PARMS	29	X'0000001D'

MQXR2_* (Wyjście Z Odpowiedź 2)

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

MQXT_* (Identyfikatory Wyjścia)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_* (Wyjście z wartości obszaru użytkownika)

Nazwa	Wartość
MQXUA_NONE	X'00...00' (16 zer)
MQXUA_NONE_ARRAY	'\0', '\0', ... (16 zer)

MQXWD_* (Wyjdź ze struktury deskryptora oczekiwania)

Nazwa	Struktura
MQXWD_STRUC_ID	"XWD¬"
Tablica MQXWD_STRUC_ID_ARRAY	'X', 'W', 'D', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXWD_VERSION_1	1	X'00000001'

MQZAC_* (Struktura kontekstu aplikacji)

Nazwa	Struktura
MQZAC_STRUC_ID	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z', 'A', 'C', '¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_* (struktura danych uprawnień)

Nazwa	Struktura
MQZAD_STRUC_ID	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_* (typy obiektów usług instalowalnych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAET_NONE	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
MQZAET_GROUP	2	X'00000002'
MQZAET_UNKNOWN	3	X'00000003'

MQZAO_* (autoryzacje usług instalacyjnych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAO_CONNECT	1	X'00000001'
MQZAO_PRZEGŁĄDANIE	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_OUTPUT	8	X'00000008'
MQZAO_ZAPYTANIE_O	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_PUBLIKOWANIE	2048	X'00000800'
MQZAO_SUBSKRYPCJA	4096	X'00001000'
MQZAO_WZNOWIENIE	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
MQZAO_CREATE	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAO_DISPLAY	262144	X'00040000'
ZMIANA MQZAO_CHANGE	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTORYZACJA	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_* (instalowalna wersja interfejsu usługi usług)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAS_VERSION_1	1	X'00000001'
MQZAS_VERSION_2	2	X'00000002'
MQZAS_VERSION_3	3	X'00000003'
MQZAS_VERSION_4	4	X'00000004'
MQZAS_VERSION_5	5	X'00000005'
MQZAS_VERSION_6	6	X'00000006'

MQZAT_* (typy uwierzytelniania)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAT_INITIAL_CONTEXT	0	X'00000000'
MQZAT_CHANGE_CONTEXT,	1	X'00000001'

MQZCI_* (indykator kontynuacji usług instalowalnych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_* (Struktura danych jednostki)

Nazwa	Struktura
MQZED_STRUC_ID	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_* (struktura wolnych parametrów)

Nazwa	Struktura
MQZFP_STRUC_ID	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZFP_VERSION_1	1	X'00000001'
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_* (Struktura kontekstu tożsamości)

Nazwa	Struktura
MQZIC_STRUC_ID	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy pusty znak.

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZIC_VERSION_1	1	X'00000001'
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_* (identyfikatory funkcji dla usług)

Identyfikatory funkcji wspólne dla wszystkich usług

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

Identyfikatory funkcji dla usługi uprawnień

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZID_INIT_AUTHORITY	0	X'00000000'
MQZID_TERM_AUTHORITY,	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY,	3	X'00000003'
MQZID_DELETE_AUTHORITY,	4	X'00000004'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
UPRAWNIENIE 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'

Identyfikatory funkcji dla usługi nazw

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Identyfikatory funkcji dla usługi Userid

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZID_INIT_USERID	0	X'00000000'
ID_UŻYTKOWNIKA MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_USERID	2	X'00000002'

MQZIO_* (Opcje Inicjowania Usług Instalacyjnych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECONDARY	1	X'00000001'

MQZNS_* (nazwa wersji interfejsu usługi)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZNS_VERSION_1	1	X'00000001'

MQZSE_* (indykator uruchamiania usług instalowalnych-indyktor liczby uruchamianej usługi)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZSE_START	1	X'00000001'

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZSE_CONTINUE	0	X'00000000'

MQZSL_* (Indykator Selektora Usług Instalacyjnych)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZSL_NOT_RETURNED	0	X'00000000'
MQZSL_RETURNED	1	X'00000001'

MQZTO_* (opcje zakończenia instalowanych usług)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECONDARY	1	X'00000001'

MQZUS_* (wersja interfejsu usługi Userid)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZUS_VERSION_1	1	X'00000001'

Typy danych używane w interfejsie MQI

Informacje na temat typów danych, które mogą być używane w interfejsie kolejki komunikatów (Message Queue Interface-MQI). Opisy, pola i deklaracje języków dla odpowiednich języków z każdym typem danych.

Wprowadzenie typów danych używanych w interfejsie MQI

W tej sekcji przedstawiono typy danych używane w interfejsie MQI, a także niektóre wskazówki dotyczące korzystania z nich w obsługiwanych językach programowania.

Elementarne typy danych

Ta sekcja zawiera informacje na temat typów danych używanych w interfejsie MQI (lub w funkcjach wyjścia). Opisano je szczegółowo, a następnie przedstawiono przykłady deklarowania elementarnych typów danych w obsługiwanych językach programowania w następujących tematach.

Typy danych używane w interfejsie MQI (lub w funkcjach wyjścia) są następujące:

- Elementarne typy danych, lub
- Agregaty elementarnych typów danych (tablice lub struktury)

Następujące elementarne typy danych są używane w interfejsie MQI (lub w funkcjach wyjścia):

Nazwa typu danych elementarnych	Typ danych	Opis
MQBOOL	wartość boolowska	Typ danych MQBOOL reprezentuje wartość boolową. Wartość 0 oznacza wartość false. Każda inna wartość reprezentuje wartość true. Obiekt MQBOOL musi być wyrównany w taki sposób, aby był zgodny z typem danych MQLONG.

Nazwa typu danych elementarnych	Typ danych	Opis
MQBYTE	Bajt	Typ danych MQBYTE reprezentuje jeden bajt danych. Żadna konkretna interpretacja nie jest umieszczana na bajcie; jest traktowana jako ciąg bitów, a nie jako liczba binarna lub znakowa. Specjalne wyrównanie nie jest wymagane. Gdy dane MQBYTE są wysyłane między menedżerami kolejek, które używają różnych zestawów znaków lub kodowań, dane produktu MQBYTE <i>nie</i> są przekształcane w żaden sposób. Pola <i>MsgId</i> i <i>CorrelId</i> w strukturze MQMD są podobne do tego. Tablica zmaterializowana MQBYTE jest czasami używana do reprezentowania obszaru pamięci głównej, który nie jest znany menedżerowi kolejek. Na przykład obszar może zawierać dane komunikatu aplikacji lub strukturę. Wyrównanie graniczne tego obszaru musi być zgodne z charakterem zawartych w nim danych. W języku programowania C każdy typ danych może być używany dla parametrów funkcji, które są wyświetlane jako tablice MQBYTE. Dzieje się tak dlatego, że takie parametry są zawsze przekazywane przez adres, a w języku C parametr funkcji jest zadeklarowany jako wskaźnik-do-void.

Nazwa typu danych elementarnych	Typ danych	Opis
MQBYTEN	łańcuch n bajtów	Każdy typ danych MQBYTEN reprezentuje łańcuch n bajtów, gdzie n może przyjmować dowolną z następujących wartości: 8, 16, 24, 32, 40 lub 128. Każdy bajt jest opisany przez typ danych MQBYTE. Specjalne wyrównanie nie jest wymagane. Jeśli dane w łańcuchu bajtowym są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełnione wartościami pustymi w celu wypełnienia łańcucha. Gdy menedżer kolejek zwraca łańcuchy bajtowe do aplikacji (na przykład w wywołaniu MQGET), bloki menedżera kolejek z wartościami pustymi są zdefiniowane na podstawie długości określonej długości łańcucha. Stałe nazwane są dostępne w celu zdefiniowania długości pól łańcucha bajtów. Są one wymienione w sekcji “Stałe” na stronie 61.

Nazwa typu danych elementarnych	Typ danych	Opis
MQCHAR	Znak	Typ danych MQCHAR reprezentuje znak jednobajtowy lub jeden bajt o dwubajtowym lub wielobajtowym znaku. Specjalne wyrównanie nie jest wymagane. Gdy dane MQCHAR są wysyłane między menedżerami kolejek, które używają różnych zestawów znaków lub kodowań, dane MQCHAR zwykle wymagają konwersji, aby dane były interpretowane poprawnie. Menedżer kolejek wykonuje to automatycznie dla danych MQCHAR w strukturze MQMD. Konwersja danych MQCHAR w danych komunikatu aplikacji jest sterowana za pomocą opcji MQGMO_CONVERT określonej w wywołaniu MQGET. Więcej szczegółów można znaleźć w opisie tej opcji w produkcie “MQGMO-Opcje Get-message” na stronie 367.

Nazwa typu danych elementarnych	Typ danych	Opis
MQCHARn	łańcuch znaków n	Każdy typ danych MQCHARn reprezentuje łańcuch znaków n znaków, gdzie n może przyjmować dowolną z następujących wartości: 4, 8, 12, 20, 28, 32, 48, 64, 128 lub 256. Każdy znak jest opisany przez typ danych MQCHAR. Specjalne wyrównanie nie jest wymagane. Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełniane spacjami, aby wypełnić łańcuch. W niektórych przypadkach znak o kodzie zero może być używany do przedwczesnego zakończenia łańcucha, zamiast dopełniania odstępami; znak o kodzie zero i znaki następujące po nim są traktowane jako odstępy, aż do długości określonej długości łańcucha. Miejsca, w których można użyć wartości NULL, są identyfikowane w opisach wywołania i typu danych. Gdy menedżer kolejek zwraca łańcuchy znaków do aplikacji (na przykład w wywołaniu MQGET), menedżer kolejek zawsze podkłada odstępy do zdefiniowanej długości łańcucha; menedżer kolejek nie używa znaku o kodzie zero do odkodowania łańcucha. Dostępne są stałe nazwane, które definiują długości pól łańcucha znaków i są wymienione w "Stale" na stronie 61.

Nazwa typu danych elementarnych	Typ danych	Opis
MQFLOAT32	32-bitowa liczba zmiennopozycyjna	Typ danych MQFLOAT32 jest 32-bitową liczbą zmiennopozycyjną reprezentowaną przy użyciu standardowego formatu zmiennopozycyjnego IEEE. Wartość MQFLOAT32 musi być wyrównana w 4-bajtowej granicy. Użycie komendy MQFLOAT32 w języku C na serwerze z/OS wymaga użycia opcji kompilatora FLOAT (IEEE). Użycie komendy MQFLOAT32 w języku COBOL jest ograniczone do kompilatorów obsługujących liczby zmiennopozycyjne w formacie IEEE. Może to wymagać użycia opcji kompilatora FLOAT (NATIVE).
MQFLOAT64	64-bitowa liczba zmiennopozycyjna	Typ danych MQFLOAT64 jest 64-bitową liczbą zmiennopozycyjną reprezentowaną przy użyciu standardowego formatu zmiennopozycyjnego IEEE. Wartość MQFLOAT64 musi być wyrównana w 8-bajtowej granicy. Użycie komendy MQFLOAT64 w języku C na serwerze z/OS wymaga użycia opcji kompilatora FLOAT (IEEE). Użycie komendy MQFLOAT64 w języku COBOL jest ograniczone do kompilatorów obsługujących liczby zmiennopozycyjne w formacie IEEE. Może to wymagać użycia opcji kompilatora FLOAT (NATIVE).

Nazwa typu danych elementarnych	Typ danych	Opis
MQHCONFIG	Uchwyt konfiguracji	Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla określonej usługi instalowalnej. Uchwyt konfiguracji musi być wyrównany względem jego naturalnej granicy. Aplikacje nie mogą opierać się na formacie danych przechowywanych w tym uchwycie. Jeśli ta wartość jest poprawna, jej wartość ma być użyteczna w dalszych wywołaniach MQI, ale nie ma znaczenia, że ma to znaczenie oprócz tego celu.
MQHCONN	Uchwyt połączenia	Typ danych MQHCONN reprezentuje uchwyt połączenia, tj. połączenie z określonym menedżerem kolejek. Uchwyt połączenia musi być wyrównany na granicy 4-bajtowej. Aplikacje nie mogą opierać się na formacie danych przechowywanych w tym uchwycie. Jeśli ta wartość jest poprawna, jej wartość ma być użyteczna w dalszych wywołaniach MQI, ale nie ma znaczenia, że ma to znaczenie oprócz tego celu.
MQHMSG	uchwyt komunikatu	Typ danych MQHMSG reprezentuje uchwyt komunikatu, który zapewnia dostęp do komunikatu. Uchwyt komunikatu musi być wyrównany na 8-bajtowej granicy. Aplikacje nie mogą opierać się na formacie danych przechowywanych w tym uchwycie. Jeśli ta wartość jest poprawna, jej wartość ma być użyteczna w dalszych wywołaniach MQI, ale nie ma znaczenia, że ma to znaczenie oprócz tego celu.

Nazwa typu danych elementarnych	Typ danych	Opis
MQHOBJ	Uchwył obiektu	Typ danych MQHOBJ reprezentuje uchwyt obiektu, który daje dostęp do obiektu. Uchwyt obiektu musi być wyrównany na granicy 4-bajtowej. Aplikacje nie mogą opierać się na formacie danych przechowywanych w tym uchwycie. Jeśli ta wartość jest poprawna, jej wartość ma być użyteczna w dalszych wywołaniach MQI, ale nie ma znaczenia, że ma to znaczenie oprócz tego celu.
MQINT8	8-bitowa liczba całkowita ze znakiem	Typ danych MQINT8 jest 8-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -128 do +127, chyba że kontekst został ograniczony przez kontekst.
MQINT16	16-bitowa liczba całkowita ze znakiem	Typ danych MQINT16 jest 16-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -32 768 do +32 767, chyba że kontekst został ograniczony przez kontekst. Wartość MQINT16 musi być wyrównana do granicy dwubajtowej.
MQINT32	32-bitowa liczba całkowita ze znakiem	Typ danych MQINT32 to 32-bitowa binarna liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, chyba że kontekst został ograniczony przez kontekst. Zapoznaj się z definicją MQLONG.

Nazwa typu danych elementarnych	Typ danych	Opis
MQINT64	64-bitowa liczba całkowita ze znakiem	Typ danych MQINT64 jest 64-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, o ile nie jest to inaczej ograniczone przez kontekst. W przypadku języka COBOL poprawny zakres jest ograniczony do -999 999 999 999 999 999 do +999 999 999 999 999 999. 64-bitowa liczba całkowita musi być wyrównana w 8-bajtowej granicy.
MQLONG	32-bitowa liczba całkowita ze znakiem	Typ danych MQLONG jest 32-bitową binarną liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, chyba że kontekst został ograniczony przez kontekst. W przypadku języka COBOL, poprawny zakres jest ograniczony do -999 999 999 do +999 999 999. Wartość MQLONG musi być wyrównana na granicy 4-bajtowej.
MQPID	Identyfikator procesu	Identyfikator procesu IBM MQ . Jest to ten sam identyfikator, który jest używany w danych śledzenia MQ i zrzutach FFST™ , ale może być inny niż identyfikator procesu systemu operacyjnego.

Nazwa typu danych elementarnych	Typ danych	Opis
MQPTR	Wskaźnik	Typ danych MQPTR to adres danych dowolnego typu. Wskaźnik musi być wyrównany względem jego naturalnej granicy; jest to 16-bajtowa granica na IBM i, a 8-bajtowa granica na innych platformach. Niektóre języki programowania obsługują wskaźniki o określonym typie. W kilku przypadkach interfejs MQI również korzysta z tych elementów (na przykład PMQCHAR i PMQLONG w języku programowania C).
Identyfikator MQTID	Identyfikator wątku	Identyfikator wątku IBM MQ . Jest to ten sam identyfikator, który jest używany w danych śledzenia MQ i zrzutach FFST™ , ale może być inny niż identyfikator wątku systemu operacyjnego.
MQUINT8	8-bitowa liczba całkowita bez znaku	Typ danych MQUINT8 jest 8-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +255, chyba że kontekst jest ograniczony przez kontekst.
MQUINT16	16-bitowa liczba całkowita bez znaku	Typ danych MQUINT16 jest 16-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +65 535, chyba że kontekst jest ograniczony przez kontekst. Wartość MQUINT16 musi być wyrównana do granicy dwubajtowej.
MQUINT32	32-bitowa liczba całkowita bez znaku	Typ danych MQUINT32 jest 32-bitową, niepodpisaną binarną liczbą całkowitą. Zapoznaj się z definicją MQULONG.

Nazwa typu danych elementarnych	Typ danych	Opis
MQINT64	64-bitowa liczba całkowita bez znaku	Typ danych MQINT64 jest 64-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +18 446 744 073 709 551 615, chyba że kontekst został ograniczony przez kontekst. W przypadku języka COBOL, poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999 999 999. 64-bitowa liczba całkowita musi być wyrównana w 8-bajtowej granicy.
MQULONG	32-bitowa liczba całkowita bez znaku	Typ danych MQULONG jest 32-bitową, niepodpisaną binarną liczbą całkowitą, która może przyjmować dowolną wartość z zakresu od 0 do + 4 294 967 294, chyba że kontekst został ograniczony przez kontekst. W przypadku języka COBOL, poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999. Wartość MQULONG musi być wyrównana na granicy 4-bajtowej.
PMQACH	Wskaźnik	Wskaźnik do struktury danych typu MQACH
PMQAIR	Wskaźnik	Wskaźnik do struktury danych typu MQAIR
PMQAXC	Wskaźnik	Wskaźnik do struktury danych typu MQAXC
PMQAXP	Wskaźnik	Wskaźnik do struktury danych typu MQAXP
PMQBMHO	Wskaźnik	Wskaźnik do struktury danych typu MQBMHO
PMQBO	Wskaźnik	Wskaźnik do struktury danych typu MQBO
PMQBOOL	Wskaźnik	Wskaźnik do danych typu MQBOOL
PMQBYTE	Wskaźnik	Wskaźnik do danych typu MQBYTE
PMQBYTEN	Wskaźnik	Wskaźnik do danych typu MQBYTEN, gdzie n może mieć wartość 8, 16, 24, 32, 40, 128

Nazwa typu danych elementarnych	Typ danych	Opis
PMQCBC	Wskaźnik	Wskaźnik do struktury danych typu MQCBC
PMQCBD	Wskaźnik	Wskaźnik do struktury danych typu MQCBD
PMQCHAR	Wskaźnik	Wskaźnik do danych typu MQCHAR
PMQCHARN	Wskaźnik	Wskaźnik do typu danych MQCHARN, gdzie n może mieć wartość 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCHARV	Wskaźnik	Wskaźnik do struktury danych typu MQCHARV
PMQCIH	Wskaźnik	Wskaźnik do struktury danych typu MQCIH
PMQCMHO	Wskaźnik	Wskaźnik do struktury danych typu MQCMHO
PMQCNO	Wskaźnik	Wskaźnik do struktury danych typu MQCNO
PMQCSP	Wskaźnik	Wskaźnik do struktury danych typu MQCSP
PMQCTLO	Wskaźnik	Wskaźnik do struktury danych typu MQCTLO
PMQDH	Wskaźnik	Wskaźnik do struktury danych typu MQDH
PMQDHO	Wskaźnik	Wskaźnik do struktury danych typu MQDHO
PMQDLH	Wskaźnik	Wskaźnik do struktury danych typu MQDLH
PMQDMHO	Wskaźnik	Wskaźnik do struktury danych typu MQDMHO
PMQDMPO	Wskaźnik	Wskaźnik do struktury danych typu MQDMPO
PMQEPH	Wskaźnik	Wskaźnik do struktury danych typu MQEPH
PMQFLOAT32	Wskaźnik	Wskaźnik do struktury danych typu MQFLOAT32
PMQFLOAT64	Wskaźnik	Wskaźnik do struktury danych typu MQFLOAT64
PMQFUNC	Wskaźnik	Wskaźnik do funkcji
PMQGMO	Wskaźnik	Wskaźnik do struktury danych typu MQGMO
PMQHCONFIG	Wskaźnik	Wskaźnik do danych typu MQHCONFIG

Nazwa typu danych elementarnych	Typ danych	Opis
PMQHCONN	Wskaźnik	Wskaźnik do danych typu MQHCONN
PMQHMSG	Wskaźnik	Wskaźnik do danych typu MQHMSG
PMQHOBJ	Wskaźnik	Wskaźnik do danych typu MQHOBJ
PMQIIH	Wskaźnik	Wskaźnik do struktury danych typu MQIIH
PMQIMPO	Wskaźnik	Wskaźnik do struktury danych typu MQIMPO
PMQINT8	Wskaźnik	Wskaźnik do danych typu MQINT8
PMQINT16	Wskaźnik	Wskaźnik do danych typu MQINT16
PMQINT32	Wskaźnik	Wskaźnik do danych typu MQINT32
PMQINT64	Wskaźnik	Wskaźnik do danych typu MQINT64
PMQLONG	Wskaźnik	Wskaźnik do danych typu PMQLONG
PMQMD	Wskaźnik	Wskaźnik do struktury typu MQMD
PMQMDE	Wskaźnik	Wskaźnik do struktury danych typu MQMDE
PMQMD1	Wskaźnik	Wskaźnik do struktury danych typu MQMD1
PMQMD2	Wskaźnik	Wskaźnik do struktury danych typu MQMD2
PMQMHBO	Wskaźnik	Wskaźnik do struktury danych typu MQMHBO
PMQOD	Wskaźnik	Wskaźnik do struktury danych typu MQOD
PMQOR	Wskaźnik	Wskaźnik do struktury danych typu MQOR
PMQPD	Wskaźnik	Wskaźnik do struktury danych typu MQPD
PMQPID	Wskaźnik	Wskaźnik do identyfikatora procesu
PMQMD	Wskaźnik	Wskaźnik do struktury danych typu MQMD
PMQPMO	Wskaźnik	Wskaźnik do struktury danych typu MQPMO

Nazwa typu danych elementarnych	Typ danych	Opis
PMQPTR	Wskaźnik	Wskaźnik do danych typu MQPTR
PMQRFH	Wskaźnik	Wskaźnik do struktury danych typu MQRFH
PMQRFH2	Wskaźnik	Wskaźnik do struktury danych typu MQRFH2
PMQRMH	Wskaźnik	Wskaźnik do struktury danych typu MQRMH
PMQRR	Wskaźnik	Wskaźnik do struktury danych typu MQRR
PMQSCO	Wskaźnik	Wskaźnik do struktury danych typu MQSCO
PMQSD	Wskaźnik	Wskaźnik do struktury danych typu MQSD
PMQSMPO	Wskaźnik	Wskaźnik do struktury danych typu MQSMPO
PMQSRO	Wskaźnik	Wskaźnik do struktury danych typu MQSRO
PMSSTS	Wskaźnik	Wskaźnik do struktury danych typu MQSTS
Identyfikator PMQTID	Wskaźnik	Wskaźnik do identyfikatora wątku
PMQTM	Wskaźnik	Wskaźnik do struktury danych typu MQTM
PMQTC2	Wskaźnik	Wskaźnik do struktury danych typu MQTC2
PMQUINT8	Wskaźnik	Wskaźnik do typu danych MQUINT8
PMQUINT16	Wskaźnik	Wskaźnik do typu danych MQUINT16
PMQUINT32	Wskaźnik	Wskaźnik do typu danych MQUINT32
PMQUINT64	Wskaźnik	Wskaźnik do typu danych MQUINT64
PMQULONG	Wskaźnik	Wskaźnik do typu danych MQULONG
PMQVOID	Wskaźnik	
PMQWIH	Wskaźnik	Wskaźnik do struktury danych typu MQWIH
PMQXQH	Wskaźnik	Wskaźnik do struktury danych typu MQXQH

C deklaracje

Typ danych	Reprezentacja
MQBOOL	<pre>typedef MQLONG MQBOOL;</pre>
MQBYTE	<pre>typedef unsigned char MQBYTE;</pre>
MQBYTE8	<pre>typedef MQBYTE MQBYTE8[8];</pre>
MQBYTE16	<pre>typedef MQBYTE MQBYTE16[16];</pre>
MQBYTE24	<pre>typedef MQBYTE MQBYTE24[24];</pre>
MQBYTE32	<pre>typedef MQBYTE MQBYTE32[32];</pre>
MQBYTE40	<pre>typedef MQBYTE MQBYTE40[40];</pre>
MQCHAR	<pre>typedef char MQCHAR;</pre>
MQCHAR4	<pre>typedef MQCHAR MQCHAR4[4];</pre>
MQCHAR8	<pre>typedef MQCHAR MQCHAR8[8];</pre>
MQCHAR12	<pre>typedef MQCHAR MQCHAR12[12];</pre>
MQCHAR20	<pre>typedef MQCHAR MQCHAR20[20];</pre>
MQCHAR28	<pre>typedef MQCHAR MQCHAR28[28];</pre>
MQCHAR32	<pre>typedef MQCHAR MQCHAR32[32];</pre>
MQCHAR48	<pre>typedef MQCHAR MQCHAR48[48];</pre>
MQCHAR64	<pre>typedef MQCHAR MQCHAR64[64];</pre>
MQCHAR128	<pre>typedef MQCHAR MQCHAR128[128];</pre>

Typ danych	Reprezentacja
MQCHAR256	<pre>typedef MQCHAR MQCHAR256[256];</pre>
MQFLOAT32	<pre>typedef float MQFLOAT32;</pre>
MQFLOAT64	<pre>typedef double MQFLOAT64;</pre>
MQHCONFIG	<pre>typedef void MQPOINTER MQHCONFIG;</pre>
MQHCONN	<pre>typedef MQLONG MQHCONN;</pre>
MQHOBJ	<pre>typedef MQLONG MQHOBJ;</pre>
MQINT8	<pre>typedef signed char MQINT8;</pre>
MQINT16	<pre>typedef short MQINT16;</pre>
MQINT64	W 64-bitowym systemie Systemy UNIX: <pre>typedef long;</pre> W 32-bitowych systemach AIX, Solaris HP-UX: <pre>typedef int64_t;</pre> W systemach IBM i, Linux z/OS: <pre>typedef long long;</pre> wł.Windows: <pre>typedef _int64;</pre>
MQLONG	wł.IBM i: <pre>typedef long MQLONG;</pre> inne platformy: <pre>if defined(MQ_64_BIT) typedef int MQLONG; else typedef long MQLONG;</pre>

Typ danych	Reprezentacja
MQPID	<pre>typedef MQLONG MQPID;</pre>
MQPTR	<pre>typedef void MQPOINTER MQPTR;</pre>
Identyfikator MQTID	<pre>typedef MQLONG MQTID;</pre>
MQUINT8	<pre>typedef unsigned char MQUINT8;</pre>
MQUINT16	<pre>typedef unsigned short MQUINT16;</pre>
MQUINT64	W 64-bitowym systemie Systemy UNIX: <pre>typedef unsigned long;</pre> W 32-bitowych systemach AIX, Solarisi HP-UX: <pre>typedef uint64_t;</pre> W systemach IBM i, Linux i z/OS: <pre>typedef unsigned long long;</pre> wł.Windows: <pre>typedef unsigned _int64;</pre>
MQULONG	wł.IBM i: <pre>typedef unsigned long MQULONG;</pre> inne platformy: <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>

Typ danych	Reprezentacja
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>
PMQCHAR128	<pre>typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];</pre>
PMQCHAR256	<pre>typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];</pre>

Typ danych	Reprezentacja
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 MQHOBJS MQPOINTER PMQHOBJS;</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>

Typ danych	Reprezentacja
PMQMDE	<code>typedef MQMDE MQPOINTER PMQMDE;</code>
PMQOD	<code>typedef MQOD MQPOINTER PMQOD;</code>
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 MQTMC2[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>
PPMQBAJT	<code>typedef PMQBYTE MQPOINTER PPMQBYTE;</code>

Typ danych	Reprezentacja
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>
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>
Gdzie defined(MQ_64_BIT) oznacza platformę 64-bitową.	

Opis zmiennej makra MQPOINTER znajduje się w sekcji [“Typy danych” na stronie 267](#).

Deklaracje języka COBOL

Typ danych	Reprezentacja
MQBOOL	<code>PIC S9(9) BINARY</code>
MQBYTE	<code>PIC X</code>
MQBYTE8	<code>PIC X(8)</code>

Typ danych	Reprezentacja
MQBYTE16	PIC X(16)
MQBYTE24	PIC X(24)
MQBYTE32	PIC X(32)
MQBYTE40	PIC X(40)
MQCHAR	PIC X
MQCHAR4	PIC X(4)
MQCHAR8	PIC X(8)
MQCHAR12	PIC X(12)
MQCHAR20	PIC X(20)
MQCHAR28	PIC X(28)
MQCHAR32	PIC X(32)
MQCHAR48	PIC X(48)
MQCHAR64	PIC X(64)
MQCHAR128	PIC X(128)
MQCHAR256	PIC X(256)
MQFLOAT32	USAGE COMP-1
MQFLOAT64	USAGE COMP-2

Typ danych	Reprezentacja
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

Deklaracje PL/I

Język PL/I jest obsługiwany w systemie z/OS.

Typ danych	Reprezentacja
MQBOOL	fixed bin(31)
MQBYTE	char(1)
MQBYTE8	char(8)
MQBYTE16	char(16)

Typ danych	Reprezentacja
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)

Typ danych	Reprezentacja
MQHOBJ	fixed bin(31)
MQINT8	fixed bin(7)
MQINT16	fixed bin(15)
MQINT64	fixed bin(63)
MQLONG	fixed bin(31)
MQPTR	pointer
MQUINT8	fixed bin(8)
MQUINT16	fixed bin(16)
MQUINT64	fixed bin(64)
MQULONG	fixed bin(32)

Deklaracje asemblera System/390

Asembler System/390 jest obsługiwany tylko w systemie z/OS .

Typ danych	Reprezentacja
MQBOOL	DS F
MQBYTE	DS XL1
MQBYTE8	DS XL8
MQBYTE16	DS XL16
MQBYTE24	DS XL24

Typ danych	Reprezentacja
MQBYTE32	DS XL32
MQBYTE40	DS XL40
MQCHAR	DS CL1
MQCHAR4	DS CL4
MQCHAR8	DS CL8
MQCHAR12	DS CL12
MQCHAR20	DS CL20
MQCHAR28	DS CL28
MQCHAR32	DS CL32
MQCHAR48	DS CL48
MQCHAR64	DS CL64
MQCHAR128	DS CL128
MQCHAR256	DS CL256
MQFLOAT32	DS EB
MQFLOAT64	DS DB
MQHCONN	DS F
MQHOBJ	DS F

Typ danych	Reprezentacja
MQINT8	DS XL1
MQINT16	DS H
MQINT64	DS D
MQLONG	DS F
MQPTR	DS F
MQUINT8	DS XL1
MQUINT16	DS H
MQUINT64	DS D
MQULONG	DS F

Typy danych struktury-wprowadzenie

W tej sekcji przedstawiono typy danych struktury używane w interfejsie MQI. Same typy danych struktury są opisane w kolejnych sekcjach.

Podsumowanie

W poniższych tabelach podsumowane są typy danych struktury używane w interfejsie MQI.

Tabela 7. Typy danych struktury używane w wywołaniach MQI (lub funkcjach wyjścia):		
Struktura	Opis	Wywołania, w których użyto
MQACH	Nagłówek łańcucha wyjścia funkcji API	
MQAIR	Rekord informacji uwierzytelniających	MQCONN
MQAXC	Kontekst wyjścia funkcji API	
MQAXP	Parametr wyjścia funkcji API	
MQBMHO	Opcje uchwytu buforu do komunikatu	MQBUFMH
MQBO	Opcje rozpoczęcia	MQBEGIN
MQCBD	Deskryptor wywołania zwrotnego	MQCB

Tabela 7. Typy danych struktury używane w wywołaniach MQI (lub funkcjach wyjścia): (kontynuacja)

Struktura	Opis	Wywołania, w których użyto
MQCBO	Opcje tworzenia worków	Torba mqCreate
MQCHARV	Łańcuch o zmiennej długości	MQINQMP
MQCNO	Opcje połączenia	MQCONN
MQCSP	Parametry bezpieczeństwa	MQCONN
MQCTLO	Opcje wywołania zwrotnego	MQCTL
MQDMPO	Opcje usuwania właściwości komunikatu	MQDLTMP
MQGMO	Opcje get-message	MQGet
MQIMPO	Zapytaj o opcje właściwości komunikatu	MQINQMP
MQMD	deskryptor komunikatu	MQBUFMH , MQMHBUF , MQCB , MQGET , MQPUT , MQPUT1
MQMHBO	Uchwyt komunikatu do opcji buforu	MQMHBUF
MQOD	deskryptor obiektu	MQOPEN , MQPUT1
MQOR	Rekord obiektu	MQOPEN , MQPUT1
MQPD	Deskryptor właściwości	MQSETMP
MQPMO	Opcje put-message	MQPUT , MQPUT1
MQPMR	Rekord komunikatu umieszczonego	MQPUT , MQPUT1
MQRR	Rekord odpowiedzi	MQOPEN , MQPUT , MQPUT1
MQSCO	Opcje konfiguracji protokołu SSL	MQCONN
MQSD	Deskryptor subskrypcji	MQSUB
MQSMPO	Opcja ustawiania właściwości komunikatu	MQSETMP
MQSRO	Opcje żądania subskrypcji	MQSUBRQ
MQSTS	Struktura raportowania statusu	MQSTAT

Tabela 8. Typy danych struktury używane w danych komunikatu:

Struktura	Opis
MQCIH ,	Nagłówek informacji CICS
MQCFH	Nagłówek PCF
MQEPH	Osadzony nagłówek PCF
MQDH	Nagłówek dystrybucji
MQDLH	Nagłówek martwego listu (niedostarczone wiadomości)
MQIIH .	Nagłówek informacji IMS

Tabela 8. Typy danych struktury używane w danych komunikatu: (kontynuacja)

Struktura	Opis
<u>MQMDE</u>	Rozszerzenie deskryptora komunikatu
<u>MQRFH</u> ,	Reguły i nagłówki formatowania
<u>MQRFH2</u>	Reguły i nagłówki formatowania 2
<u>MQRMH</u>	Nagłówek komunikatu odwołania
<u>MQTM</u>	komunikat wyzwalacza
<u>MQTM2</u>	Komunikat wyzwalacza (format znakowy 2)
<u>MQWIH</u>	Nagłówek informacji o pracy
<u>MQXQH</u>	Nagłówek kolejki transmisji

Uwaga: Struktura MQDXP (parametr wyjścia konwersji danych) jest opisana w podręczniku [“Konwersja danych”](#) na stronie 919, wraz z powiązanymi wywołaniami konwersji danych.

Reguły dla typów danych struktury

Języki programowania różnią się w zależności od poziomu obsługi struktur, a niektóre reguły i konwencje są przyjmowane w celu spójnego odwzorowania struktur MQI w każdym języku programowania:

- Struktury muszą być dopasowane do ich naturalnych granic.
 - Większość struktur MQI wymaga wyrównania 4-bajтового.
 - W systemie IBM istruktury zawierające wskaźniki wymagają wyrównania 16-bajтового. Są to: MQCNO, MQOD, MQPMO.
- Każde pole w strukturze musi być wyrównane na jego naturalnej granicy.
 - Pola z typami danych, które są zgodne z typem MQLONG, muszą być wyrównane w granicach 4-bajtowych.
 - Pola z typami danych, które są zgodne z MQPTR, muszą być wyrównane do 16-bajtowych granic w systemach IBM i 4-bajtowych granicach w innych środowiskach.
 - Pozostałe pola są wyrównywane w zakresie 1-bajtowym.
- Długość struktury musi być wielokrotnością jego wyrównania granicznego.
 - Większość struktur MQI ma długości, które są wielokrotnością 4 bajtów.
 - W systemie IBM istruktury zawierające wskaźniki mają długości, które są wielokrotnością 16 bajtów.
- W razie potrzeby należy dodać dopełnianie bajtów lub pól w celu zapewnienia zgodności z poprzednimi regułami.

Konwencje używane w opisach

Opis każdego typu danych struktury zawiera następujące elementy:

- Przegląd celu i wykorzystania struktury
- Opisy pól w strukturze, w postaci, która jest niezależna od języka programowania
- Przykłady deklarowanych struktur w każdym z obsługiwanych języków programowania

Opis każdego typu danych struktury zawiera następujące sekcje:

Nazwa struktury

Nazwa struktury, po której następuje podsumowanie pól w strukturze.

Przegląd

Krótki opis przeznaczenia i zastosowania konstrukcji.

Pola

Opisy pól. Dla każdego pola po nazwie pola występuje jego elementarny typ danych w nawiasach (). W tekście nazwy pól są wyświetlane za pomocą kursywa, na przykład *Version*.

Znajduje się tam również opis celu pola wraz z listą dowolnych wartości, jakie może podjąć pole. Nazwy stałych są wyświetlane wielkimi literami, na przykład MQGMO_STRUC_ID. Zestaw stałych o tym samym przedrostku jest wyświetlany za pomocą znaku *, na przykład: MQIA_ *.

W opisach pól używane są następujące terminy:

wejściowe,

Informacje na temat podaży są wyświetlane podczas wywoływania.

wyniki

Menedżer kolejek zwraca informacje w tym polu, gdy wywołanie zakończy się lub zakończy się niepowodzeniem.

Wejście/wyjście

Informacje są wprowadzane w polu podczas wykonywania wywołania, a menedżer kolejek zmienia informacje, gdy wywołanie zakończy się lub zakończy się niepowodzeniem.

Wartości początkowe

Tabela przedstawiana wartości początkowych dla każdego pola w plikach definicji danych dostarczonych wraz z interfejsem MQI.

Deklaracja C

Typowa deklaracja struktury w C.

Deklaracja języka COBOL

Typowa deklaracja struktury w języku COBOL.

Deklaracja PL/I

Typowa deklaracja struktury w PL/I.

Deklaracja asemblera System/390

Typowa deklaracja struktury w języku asemblera System/390 .

Wizualna deklaracja podstawowa

Typowa deklaracja struktury w Visual Basic.

programowanie w języku C

Ta sekcja zawiera informacje pomocne podczas korzystania z interfejsu MQI z języka programowania C.

Pliki nagłówkowe

Pliki nagłówkowe są udostępniane w celu ułatwienia pisania programów aplikacji C, które korzystają z interfejsu MQI.

Te pliki nagłówkowe są podsumowane w programie [Tabela 9 na stronie 266](#).

Tabela 9. Pliki nagłówkowe C	
Plik	Spis treści
CMQC	Prototypy funkcji, typy danych i stałe nazwane dla głównego interfejsu MQI
CMQXC	Prototypy funkcji, typy danych i stałe nazwane dla wyjścia konwersji danych
CMQEC	Prototypy funkcji, typy danych i stałe nazwane dla głównej struktury interfejsu MQI, wyjścia konwersji danych i punktu wejścia interfejsu (CMQEC zawiera CMQXC i CMQC.)
CMQSTRC	Funkcje, które przekształcają definicje stałej MQI w odpowiednik tekstu.
	 Ostrzeżenie: z/OS Nie dotyczy produktu z/OS.

Aby zwiększyć przenośność aplikacji, należy zakodować nazwę pliku nagłówkowego małymi literami w dyrektywie preprocesora `#include` :

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

Funkcje

Nie ma potrzeby określania wszystkich parametrów, które są przekazywane przy każdym wywołaniu funkcji przy każdym wywołaniu funkcji.

- Przekaz parametry, które są *tylko wejściowe* i typu MQHCONN, MQHOBJ lub MQLONG według wartości.
- Przekaz wszystkie pozostałe parametry według adresu.

Jeśli określony parametr nie jest wymagany, należy użyć pustego wskaźnika jako parametru w wywołaniu funkcji, w miejsce adresu danych parametru. Parametry, dla których jest to możliwe, są identyfikowane w opisach wywołań.

Żaden parametr nie jest zwracany jako wartość funkcji; w terminologii C oznacza to, że wszystkie funkcje zwracają wartość `void`.

Atrybuty tej funkcji są definiowane przez zmienną makra MQENTRY. Wartość tej zmiennej makra zależy od środowiska.

Parametry z niezdefiniowanym typem danych

Parametr *Buffer* w funkcjach MQGET, MQPUT i MQPUT1 ma niezdefiniowany typ danych. Ten parametr jest używany do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są przedstawione w przykładach C jako tablice MQBYTE. Parametry można deklarować w ten sposób, ale zwykle wygodniejsze jest zadeklarowanie ich jako konkretnej struktury, która opisuje układ danych w komunikacie. Zadeklaruj rzeczywisty parametr funkcji jako wskaźnik-do-void i określ adres dowolnego rodzaju danych jako parametr w wywołaniu funkcji.

Typy danych

Zdefiniuj wszystkie typy danych za pomocą instrukcji C `typedef`. Dla każdego typu danych zdefiniuj również odpowiedni typ danych wskaźnika. Nazwa typu danych wskaźnika to nazwa podstawowego lub strukturalnego typu danych poprzedzona literą P w celu oznaczenia wskaźnika. Zdefiniuj atrybuty wskaźnika przy użyciu zmiennej makra MQPOINTER. Wartość tej zmiennej makra zależy od środowiska. Poniżej przedstawiono sposób deklarowania typów danych wskaźników:

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

Manipulowanie łańcuchami binarnymi

Deklaruj łańcuchy danych binarnych jako jeden z typów danych MQBYTEn.

Podczas kopiowania, porównywania lub ustawiania pól tego typu, należy używać funkcji C `memcpy`, `memcmp` lub `memset`; na przykład:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,          /* set "MsgId" field to nulls */
       MQMI_NONE,               /* ...using named constant */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,       /* set "CorrelId" field to nulls */
       0x00,                   /* ...using a different method */
       sizeof(MQBYTE24));
```

Nie należy używać funkcji łańcuchowych `strcpy`, `strcmp`, `strncpy` lub `strncmp`, ponieważ te funkcje nie działają poprawnie dla danych zadeklarowanych z typami danych `MQBYTEn`.

Manipulowanie łańcuchami znaków

Gdy menedżer kolejek zwraca dane znakowe do aplikacji, menedżer kolejek zawsze dopełnia dane znakowe z odstępami do zdefiniowanej długości pola. Menedżer kolejek *nie* zwraca łańcuchów zakończonych znakiem o kodzie zero.

Dlatego przy kopiowaniu, porównywaniu lub konkatelowaniu takich łańcuchów należy użyć funkcji łańcuchowych `strncpy`, `strncmp` lub `strncat`.

Nie należy używać funkcji łańcuchowych, które wymagają, aby łańcuch został zakończony znakiem o wartości `NULL` (`strcpy`, `strcmp`, `strcat`). Nie należy również używać funkcji `strlen` do określenia długości łańcucha. Zamiast tego należy użyć funkcji `sizeof`, aby określić długość pola.

Wartości początkowe dla struktur

Pliki nagłówkowe definiują różne zmienne makra, których można użyć w celu udostępnienia początkowych wartości struktur MQ podczas deklarowania instancji tych struktur.

Te zmienne makra mają nazwy w postaci `MQxxx_DEFAULT`, gdzie `MQxxx` reprezentuje nazwę struktury. Są one używane w następujący sposób:

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

W przypadku niektórych pól znakowych (na przykład pól *StrucId*, które występują w większości struktur, lub w polu *Format* występujący w strukturze `MQMD`), MQI definiuje konkretne wartości, które są poprawne. Dla każdej z poprawnych wartości dostępne są *dwie* zmienne makra:

- Jedna zmienna makra definiuje wartość jako łańcuch o długości, z wyłączeniem niejawnych zgodności o wartości `NULL`, dokładnie określonej długości pola. Na przykład w przypadku pola *Format* w strukturze `MQMD` udostępniana jest następująca zmienna makra (– reprezentuje pusty znak):

```
#define MQFMT_STRING "MQSTR- -"
```

Tego formularza należy używać razem z funkcjami `memcpy` i `memcmp`.

- Inna zmienna makra definiuje wartość jako tablicę znaków; nazwa tej zmiennej makra jest nazwą formularza łańcucha w postaci przyrostka `_ARRAY`. Na przykład:

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

Użyj tego formularza, aby zainicjować pole podczas deklarowania instancji struktury o wartościach innych niż te, które są udostępniane przez zmienną makra `MQMD_DEFAULT`. (Nie zawsze jest to konieczne; w niektórych środowiskach można użyć postaci łańcuchowej wartości w obu sytuacjach. Można jednak użyć formularza tablicy dla deklaracji, ponieważ jest to wymagane ze względu na kompatybilność z językiem programowania C++).

Wartości początkowe dla struktur dynamicznych

Jeśli wymagana jest zmienna liczba instancji struktury, instancje są zwykle tworzone w głównej pamięci masowej uzyskanej dynamicznie przy użyciu funkcji `calloc` lub `malloc`. Aby zainicjować pola w takich strukturach, należy wziąć pod uwagę następującą technikę:

1. Zadeklaruj instancję struktury, używając odpowiedniej zmiennej makra `MQxxx_DEFAULT` w celu zainicjowania struktury. Ta instancja staje się modelem dla innych instancji:

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

Słowa kluczowe `static` lub `auto` można zakodować w deklaracji w celu nadania modelowi instancji statycznej lub dynamicznego czasu życia, zgodnie z wymaganiami.

2. Użyj funkcji `calloc` lub `malloc` , aby uzyskać pamięć masową dla dynamicznej instancji struktury:

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

3. Użyj funkcji `memcpy` , aby skopiować instancję modelu do instancji dynamicznej:

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

Użyj z języka C++

W przypadku języka programowania C + + pliki nagłówkowe zawierają następujące dodatkowe instrukcje, które są uwzględniane tylko w przypadku kompilatora C + +:

```
#ifndef __cplusplus  
extern "C" {  
#endif  
  
/* rest of header file */  
  
#ifdef __cplusplus  
}  
#endif
```

Konwencje z adnotacjami

Ta informacja przedstawia sposób wywoływania funkcji i deklarowania parametrów.

W niektórych przypadkach parametry są tablicami o wielkości, która nie jest stała. W przypadku tych wartości małe litery `n` są używane do reprezentowania stałej liczbowej. Po zakodowaniu deklaracji dla tego parametru należy zastąpić `n` wartością numeryczną wymaganą.

programowanie w języku COBOL

Ta sekcja zawiera informacje pomocne przy użyciu interfejsu MQI z języka programowania COBOL.

Kopiuj pliki

Udostępniono różne pliki COPY, które ułatwiają pisanie programów aplikacji w języku COBOL, które korzystają z interfejsu MQI. Istnieją dwa pliki zawierające nazwane stałe i dwa pliki dla każdej ze struktur.

Każda struktura jest podana w dwóch formach: postaci z wartościami początkowymi, oraz formularza bez:

- Należy użyć struktur z wartościami początkowymi w sekcji WORKING-STORAGE SECTION programu w języku COBOL. Są one zawarte w plikach COPY z przyrostkami z przyrostkiem V (dla wartości).
- Struktury bez wartości początkowych należy używać w LINKAGE SECTION programu w języku COBOL. Są one zawarte w plikach COPY z nazwami przyrostkami z literą L (dla Linkage).

Pliki COPY są podsumowywane w programie [Tabela 10 na stronie 269](#). Nie wszystkie wymienione pliki są dostępne we wszystkich środowiskach.

Tabela 10. Pliki języka COBOL COPY		
Plik (z wartościami początkowymi)	Plik (bez wartości początkowych)	Spis treści
CMQAIRV	CMQAIRL	Rekord informacji uwierzytelniających
CMQBOV	CMQBOL	Struktura opcji begin
CMQCIHV	CMQCIHL	Struktura nagłówka informacyjnego produktu CICS
CMQCN OV	CMQCNOL	Struktura opcji łączenia

Tabela 10. Pliki języka COBOL COPY (kontynuacja)

Plik (z wartościami początkowymi)	Plik (bez wartości początkowych)	Spis treści
CMQDHV	CMQDHL	Struktura nagłówka dystrybucji
CMQDLHV	CMQDLHL	Struktura nagłówka martwej litery
CMQDXPV	CMQDXPL	Struktura parametru wyjścia konwersji danych
CMQGMOV	CMQGMOL	Pobierz strukturę opcji komunikatu
CMQIIHV	CMQIIHL	Struktura nagłówka informacyjnego produktu IMS
CMQMDV	CMQMDL	Struktura deskryptora komunikatu
CMQMDEV	CMQMDEL	Struktura rozszerzenia deskryptora komunikatu
CMQMD1V	CMQMD1L	Struktura deskryptora komunikatu wersja 1
CMQODV	CMQODL	Struktura deskryptora obiektu
CMQORV	CMQORL	Struktura rekordu obiektu
CMQPMOV	CMQPMOL	Struktura opcji umieszczania komunikatów
CMQRFHV	CMQRFHL	Reguły i struktura nagłówka formatowania
CMQRFH2V	CMQRFH2L	Reguły i formatowanie struktury nagłówka w wersji 2
CMQRMHV	CMQRMHL	Struktura nagłówka komunikatu odwołania
CMQRRV	CMQRRL	Struktura rekordu odpowiedzi
CMQSCOV	KMQSCOL	Opcje konfiguracji protokołu SSL
CMQTMV	CMQTML	Struktura komunikatu wyzwalacza
CMQTMCV	CMQTMCL	Struktura komunikatu wyzwalacza (format znakowy)
CMQTM2V	CMQTM2L	Struktura komunikatu wyzwalacza (format znakowy) wersja 2
CMQWIHV	CMQWIHL	Struktura nagłówka informacji o pracy
CMQXQHV	CMQXQHL	Struktura nagłówka kolejki transmisji
CMQV	-	Stałe nazwane dla głównego interfejsu MQI
CMQXV	-	Stałe nazwane dla wyjścia konwersji danych
CMQMD2V	CMQMD2L	Struktura deskryptora komunikatu wersja 2

Struktury

W pliku COPY każda deklaracja struktury rozpoczyna się od elementu level-10 ; umożliwia to zadeklarowanie kilku instancji struktury poprzez kodowanie deklaracji level-01 , a następnie za pomocą instrukcji COPY do skopiowania w pozostałej części deklaracji struktury. Aby odwołać się do odpowiedniej instancji, należy użyć słowa kluczowego IN :

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

Wyrównaj struktury na odpowiednich granicach. W przypadku użycia instrukcji COPY w celu dołączenia struktury po elemencie, który nie jest pozycją level-01, należy upewnić się, że struktura rozpoczyna się od odpowiedniego przesunięcia od początku elementu level-01. Większość struktur MQI wymaga wyrównania 4-bajtowego. Wyjątki od tego są MQCNO, MQOD i MQPMO, które wymagają wyrównania 16-bajtowego w systemie IBM i.

W tej sekcji nazwy pól w strukturach są wyświetlane bez przedrostka. W języku COBOL nazwy pól są poprzedzane nazwą struktury, po której następuje myślnik. Jeśli jednak nazwa struktury kończy się cyfrą, co oznacza, że struktura jest drugą lub późniejszą wersją oryginalnej struktury, cyfra cyfra jest pomijana z przedrostkiem. Nazwy pól w języku COBOL są wyświetlane wielkimi literami (choć w razie potrzeby można użyć małych liter lub małych liter). Na przykład pole *MsgType* opisane w sekcji [“MQMD-deskryptor komunikatu”](#) na stronie 417 stanie się MQMD-MSGTYPE w języku COBOL.

Struktury przyrostków V są deklarowane przy użyciu wartości początkowych dla wszystkich pól; należy ustawić tylko te pola, w których ma być używana wartość inna niż podana wartość początkowa.

Wskaźniki

Niektóre struktury muszą zająć się opcjonalnymi danymi, które mogą być niecierpione ze strukturą, takimi jak rekordy MQOR i MQRR adresowane przez strukturę MQOD.

Aby rozwiązać te opcjonalne dane, struktury zawierają pola zadeklarowane za pomocą typu danych wskaźnika. Jednak język COBOL nie obsługuje wskaźnika typu danych we wszystkich środowiskach. Z tego powodu dane opcjonalne mogą być również adresowane za pomocą pól, które zawierają przesunięcie danych od początku struktury.

Aby port był używany między środowiskami, należy upewnić się, czy typ danych wskaźnika jest dostępny we wszystkich środowiskach, w których występuje. Jeśli nie jest, aplikacja musi zwracać się do opcjonalnych danych, korzystając z pól offsetowych zamiast pól wskaźnika.

W tych środowiskach, w których nie są obsługiwane wskaźniki, należy zadeklarować pola wskaźnika jako łańcuchy bajtowe odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null. Nie należy zmieniać tej wartości początkowej, jeśli używane są zmienne przesunięcia.

Stałe nazwane

W tej sekcji wyświetlane są nazwy stałych, które zawierają znak podkreślenia (_) jako część nazwy. W języku COBOL należy użyć znaku łącznika (-) w miejsce podkreślenia.

Stałe, które mają wartości łańcuchowe, używają jednego znaku cudzysłowu jako separatora łańcucha ('). W niektórych środowiskach może być konieczne podanie odpowiedniej opcji kompilatora, aby kompilator akceptowało pojedynczy cudzysłów jako ogranicznik łańcucha w miejscu podwójnego cudzysłowu.

Stałe nazwane są deklarowane w plikach COPY jako elementy level-10. Aby użyć stałych, należy jawnie zadeklarować element level-01, a następnie użyć instrukcji COPY do skopiowania w deklaracjach stałych:

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

Powyższa metoda powoduje, że stałe zajmują pamięć masową w programie, nawet jeśli nie są one przywoływane. Jeśli użytkownik uwzględni stałe w wielu oddzielnych programach w obrębie tej samej jednostki wykonywania, wówczas istnieje wiele kopii stałych, które niepotrzebnie zużywają pamięć główną. Należy unikać tego efektu przy użyciu jednej z następujących technik:

- Dodaj klauzulę GLOBAL do deklaracji level-01:

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

Powoduje to przydzielaniem pamięci tylko dla jednego zestawu stałych w obrębie jednostki wykonywania. Jednak stałe mogą być przywoływane przez dowolny program w obrębie jednostki uruchamiania, a nie tylko program, który zawiera deklarację level-01.

Uwaga: Klauzula GLOBAL nie jest obsługiwana we wszystkich środowiskach.

- Ręcznie skopiuj do każdego programu tylko te stałe, które są przywoływane przez ten program. Nie należy używać instrukcji COPY w celu skopiowania wszystkich stałych do programu.

Konwencje z adnotacjami

Te ostatnie tematy w tej sekcji pokazują, jak wywołać wywołania i zadeklarować parametry. W niektórych przypadkach parametry są tabelami lub łańcuchami znaków, których wielkość nie jest ustalona. W przypadku tych wartości małe litery n są używane do reprezentowania stałej liczbowej. Po zakodowaniu deklaracji dla tego parametru należy zastąpić n wartością numeryczną wymaganą.

Programowanie w programie System/390 assembler

Ta sekcja zawiera informacje pomocne w użyciu interfejsu MQI z języka programowania System/390 Assembler.

Makra

Dostępne są różne makra ułatwiające pisanie programów aplikacji assembler, które używają interfejsu MQI.

Istnieją dwa makra dla nazwanych stałych i jedno makro dla każdej struktury. Te pliki są podsumowane w programie Tabela 11 na stronie 272.

<i>Tabela 11. Makra asemlera</i>	
Plik	Spis treści
CMQA	Stałe nazwane (równanie) dla głównego interfejsu MQI
CMQCIHA	Struktura nagłówka informacyjnego produktu CICS
CMQCNOA	Struktura opcji łączenia
CMQDLHA	Struktura nagłówka martwej litery
CMQDXPA	Struktura parametru wyjścia konwersji danych
CMQGMOA	Pobierz strukturę opcji komunikatu
CMQIIHA	Struktura nagłówka informacyjnego produktu IMS
CMQMDA	Struktura deskryptora komunikatu
CMQMDEA	Struktura rozszerzenia deskryptora komunikatu
CMQODA	Struktura deskryptora obiektu
CMQPMOA	Struktura opcji umieszczania komunikatów
CMQRFHA	Reguły i struktura nagłówka formatowania
CMQRFH2A	Reguły i formatowanie struktury nagłówka w wersji 2
CMQRMHA	Struktura nagłówka komunikatu odwołania
CMQTMA	Struktura komunikatu wyzwacza
CMQTMCA	Struktura komunikatu wyzwacza (format znakowy) wersja 2
CMQVERA	Kontrola wersji struktury
CMQWIHA	Struktura nagłówka informacji o pracy
CMQXA	Stałe nazwane dla wyjścia konwersji danych
CMQXPA	Struktura parametru wyjścia przekraczania interfejsu API
CMQXQHA	Struktura nagłówka kolejki transmisji

Struktury

Struktury są generowane przez makra, które mają różne parametry do sterowania działaniem makra. Parametry te zostały opisane w poniższych sekcjach.

Od czasu do czasu wprowadzane są nowe wersje struktur MQ. Dodatkowe pola w nowej wersji mogą spowodować, że struktura, która wcześniej była mniejsza niż 256 bajtów, stała się większa niż 256 bajtów. Z tego powodu instrukcje dotyczące assemblera zapisu przeznaczone do kopiowania struktury produktu MQ lub ustawienia struktury MQ na wartości NULL umożliwiają poprawne działanie ze strukturami, które mogą być większe niż 256 bajtów. Alternatywnie można użyć parametru makra DCLVER lub makra CMQVERA z parametrem VERSION, aby zadeklarować konkretną wersję struktury.

Określanie nazwy struktury

Aby zadeklarować więcej niż jedną instancję struktury, makro prefikuje nazwę każdego pola w strukturze z łańcuchem określonym przez użytkownika i znakiem podkreślenia.

Używany łańcuch jest etykietą podaną w wywołaniu makra. Jeśli nie określono żadnej etykiety, do konstruowania przedrostka używana jest nazwa struktury:

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

Deklaracje struktury przedstawione w tej sekcji korzystają z domyślnego przedrostka.

Określanie formy struktury

Deklaracje struktury mogą być generowane przez makro w jednej z dwóch form, sterowanych przez parametr DSECT:

DSECT = TAK

Instrukcja assemblera DSECT jest używana do uruchamiania nowej sekcji danych. Definicja struktury jest natychmiast zgodna z instrukcją DSECT. Etykieta w wywołaniu makra jest używana jako nazwa sekcji danych. Jeśli nie określono żadnej etykiety, używana jest nazwa struktury.

DSECT = NIE

Instrukcje DC assemblera są używane do definiowania struktury w bieżącej pozycji w podprogramie. Pola są inicjowane z wartościami, które można określić, kodując odpowiednie parametry w wywołaniu makra. Pola, dla których nie są określone żadne wartości w wywołaniu makra, są inicjowane z wartościami domyślnymi.

Podana wartość musi być podana wielkimi literami. Jeśli parametr DSECT nie jest określony, przyjmowany jest DSECT=NO.

Sterowanie wersją struktury

Domyślnie makra zawsze deklarują najświeższą wersję każdej struktury.

Chociaż można użyć parametru makra VERSION w celu określenia wartości dla pola *Version* w strukturze, parametr ten definiuje wartość początkową dla pola *Version* i nie kontroluje wersji rzeczywiście zadeklarowanej struktury. Aby kontrolować wersję zadeklarowanej struktury, należy użyć parametru DCLVER:

DCLVER=CURRENT

Zadeklarowana wersja to bieżąca (najnowsza) wersja.

DCLVER=SPECIFIED

Zadeklarowana wersja jest wersją określoną przez parametr VERSION. Jeśli parametr VERSION zostanie pominięty, wartością domyślną jest wersja 1.

Jeśli zostanie określony parametr VERSION, wartość musi być samodefiniującą stałą numeryczną lub stałą nazwana dla wymaganej wersji (na przykład MQCNO_VERSION_3). Jeśli zostanie określona inna wartość, struktura zostanie zadeklarowana tak, jakby DCLVER=CURRENT została określona, nawet jeśli wartość parametru VERSION jest tłumaczona na poprawną wartość.

Podana wartość musi być podana wielkimi literami. Jeśli parametr DCLVER zostanie pominięty, użyta wartość jest pobierana z globalnej zmiennej makra MQDCLVER . Tę zmienną można ustawić za pomocą makra CMQVERA.

Deklarowanie jednej struktury osadzonej w innym

Aby zadeklarować jedną strukturę jako komponent innej struktury, należy użyć parametru NESTED :

NESTED=TAK

Deklaracja struktury jest zagnieżdżona w innym.

NESTED=NIE

Deklaracja struktury nie jest zagnieżdżona w innym.

Podana wartość musi być podana wielkimi literami. Jeśli parametr NESTED zostanie pominięty, przyjmowany jest NESTED=NO .

Określanie wartości początkowych dla pól

Należy określić wartość, która ma być używana do inicjowania pola w strukturze, kodując nazwę tego pola (bez przedrostka) jako parametr w wywołaniu makra, wraz z wymaganą wartością.

Na przykład, aby zadeklarować strukturę deskryptora komunikatu za pomocą pola *MsgType* zainicjowanego za pomocą MQMT_REQUEST, a pole *ReplyToQ* zainicjowane za pomocą łańcucha "MY_REPLY_TO_QUEUE", należy użyć następującej komendy:

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

Jeśli jako wartość w wywołaniu makra zostanie określona stała nazwana (equate), należy użyć makra CMQA w celu zdefiniowania stałej nazwanej. Nie należy ujmować wartości łańcucha znaków w pojedynczych cudzysłowach.

Sterowanie listingami

Sterowanie wyglądem deklaracji struktury w listingu asemblera za pomocą parametru LIST :

LIST = TAK

Deklaracja struktury jest wyświetlana na listingu asemblera.

LISTA = NIE

Deklaracja struktury nie jest wyświetlana na listingu asemblera.

Podana wartość musi być podana wielkimi literami. Jeśli parametr LIST zostanie pominięty, przyjmowany jest LIST=NO .

Makro CMQVERA

To makro umożliwia ustawienie wartości domyślnej, która ma być używana dla parametru DCLVER w makrach struktury. Wartość określona przez wartość CMQVERA jest używana przez makro struktury tylko wtedy, gdy parametr DCLVER jest pomijany z wywołania makra struktury. Wartość domyślna jest ustawiana poprzez kodowanie makra CMQVERA za pomocą parametru DCLVER :

DCLVER=CURRENT

Wersja domyślna jest ustawiona na bieżącą (najnowsza) wersję.

DCLVER=SPECIFIED

Wersja domyślna jest ustawiana na wersję określoną przez parametr VERSION .

Należy podać parametr DCLVER , a wartość musi być podana wielkimi literami. Wartość ustawiona przez wartość CMQVERA pozostaje wartością domyślną do czasu następnego wywołania CMQVERA lub zakończenia zespołu. Jeśli parametr CMQVERA zostanie pominięty, wartością domyślną jest DCLVER=CURRENT .

Konwencje z adnotacjami

Późniejsze sekcje przedstawiają sposób wywoływania wywołań i deklarowania parametrów. W niektórych przypadkach parametry są tablicami lub łańcuchami znakowymi o wielkości, która nie jest stała, dla której mała litera n jest używana do reprezentowania stałej numerycznej. Po zakodowaniu deklaracji dla tego parametru należy zastąpić n wartością numeryczną wymaganą.

MQAIR-rekord informacji uwierzytelniającej

Struktura MQAIR reprezentuje rekord informacji uwierzytelniających.

W poniższej tabeli podsumowano pola w strukturze.

Tabela 12. Pola w MQAIR		
Pole	Opis	Temat
StrucId	Identyfikator struktury	StrucId
Wersja	Numer wersji struktury	Wersja
Typ AuthInfo	Typ informacji uwierzytelniających	TypAuthInfo
AuthInfoConnName	Nazwa połączenia serwera CRL LDAP	AuthInfoConnName
LDAPUserNamePtr	Adres nazwy użytkownika LDAP	LDAPUserNamePtr
Przesunięcie LDAPUserName	Przesunięcie nazwy użytkownika LDAP od początku wywołania MQSCO	LDAPUserNamePrzesunięcie
Długość elementu LDAPUserName	Długość nazwy użytkownika LDAP	LDAPUserNameDługość
Hasło_LDAPPassword	Hasło dostępu do serwera LDAP	hasło_LDAPPassword
Uwaga: Pozostałe pola są ignorowane, jeśli <i>Wersja</i> jest mniejsza niż MQAIR_VERSION_2.		
OCSPResponderURL	Adres URL, z którym można się skontaktować z responderem OCSP	OCSPResponderURL

Przegląd produktu MQAIR

Struktura MQAIR umożliwia uruchomienie aplikacji jako IBM MQ MQI client w celu określenia informacji o uwierzytelniającym, który ma być używany dla połączenia klienckiego. Struktura jest parametrem wejściowym w wywołaniu MQCONN.

Dostępność: klienci AIX, HP-UX, Solaris, Linux i Windows .

Zestaw znaków i kodowanie: Dane w programie MQAIR muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one podawane przez atrybut menedżera kolejek produktu **CodedCharSetId** i atrybut MQENC_NATIVE.

Pola dla produktu MQAIR

Struktura MQAIR zawiera następujące pola: pola są opisane w **kolejności alfabetycznej** .

AuthInfoConnName (MQCHAR264)

Jest to albo nazwa hosta, albo adres sieciowy hosta, na którym działa serwer LDAP. Po tym może wystąpić opcjonalny numer portu, ujęty w nawiasy. Domyślny numer portu to 389.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_CONN_NAME_ERROR.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_AUTH_INFO_CONN_NAME_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C oraz puste znaki w innych językach programowania.

Typ AuthInfo(MQLONG)

Jest to typ informacji uwierzytelniających zawartych w rekordzie.

Wartość może być jednym z dwóch następujących parametrów:

MQAIT_CRL_LDAP

Sprawdzanie odwołań certyfikatów przy użyciu serwera LDAP.

MQAIT_OCSP

Sprawdzanie odwołań certyfikatów przy użyciu protokołu OCSP.

Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_TYPE_ERROR.

To jest pole wejściowe. Wartością początkową tego pola jest MQAIT_CRL_LDAP.

LDAPPassword (MQCHAR32)

Jest to hasło wymagane do uzyskania dostępu do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola.

Jeśli serwer LDAP nie wymaga hasła lub użytkownik pominie nazwę użytkownika LDAP, wartość *LDAPPassword* musi mieć wartość NULL lub być pusta. Jeśli nazwa użytkownika LDAP zostanie pominięta, a *LDAPPassword* nie ma wartości NULL ani nie ma wartości pustej, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_LDAP_PASSWORD_ERROR.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_LDAP_PASSWORD_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C oraz puste znaki w innych językach programowania.

Długość LDAPUserName(MQLONG)

Jest to długość w bajtach nazwy użytkownika LDAP, która jest adresowana w polu *LDAPUserNamePtr* lub *LDAPUserNameOffset*. Wartość musi być z zakresu od zera do MQ_DISTINGUISHED_NAME_LENGTH. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_LDAP_USER_NAME_LENGTH_ERR.

Jeśli używany serwer LDAP nie wymaga nazwy użytkownika, należy ustawić wartość tego pola na zero.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

LDAPUserNamePrzesunięcie (MQLONG)

Jest to przesunięcie (w bajtach) nazwy użytkownika LDAP od początku struktury MQAIR.

Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli wartość *LDAPUserNameLength* wynosi zero.

Można użyć opcji *LDAPUserNamePtr* lub *LDAPUserNameOffset*, aby określić nazwę użytkownika LDAP, ale nie obie te wartości. Szczegółowe informacje można znaleźć w opisie pola *LDAPUserNamePtr*.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

LDAPUserNamePtr (PMQCHAR)

Jest to nazwa użytkownika LDAP.

Składa się ona z nazwy wyróżniającej użytkownika, który próbuje uzyskać dostęp do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość określona w polu *LDAPUserNameLength*, należy zakończyć ją znakiem o kodzie zero lub dopełniać odstępami do długości *LDAPUserNameLength*. Pole jest ignorowane, jeśli wartość *LDAPUserNameLength* wynosi zero.

Nazwę użytkownika LDAP można podać na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *LDAPUserNamePtr*

W takim przypadku aplikacja może zadeklarować łańcuch, który jest oddzielony od struktury MQAIR, i ustawić parametr *LDAPUserNamePtr* na adres tego łańcucha.

Należy rozważyć użycie produktu *LDAPUserNamePtr* dla języków programowania, które obsługują typ danych wskaźnika w sposób przenośny dla różnych środowisk (na przykład język programowania w języku C).

- Za pomocą pola przesunięcia *LDAPUserNameOffset*

W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą strukturę MQSCO, po której następuje tablica rekordów MQAIR, po których następuje łańcuch nazwy użytkownika LDAP, a następnie ustaw *LDAPUserNameOffset* na przesunięcie odpowiedniej nazwy łańcucha nazwy od początku struktury MQAIR. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być zakwaterowana w tabeli MQLONG (najbardziej restrykcyjnym językiem programowania jest język COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

Należy rozważyć użycie produktu *LDAPUserNameOffset* dla języków programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który może nie być przenośny dla różnych środowisk (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki, należy używać tylko jednej z następujących opcji: *LDAPUserNamePtr* i *LDAPUserNameOffset*; Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_LDAP_USER_NAME_ERROR, jeśli oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

OCSPResponderURL (MQCHAR256)

W przypadku struktury MQAIR, która reprezentuje szczegóły połączenia dla modułu odpowiadającego OCSP, pole to zawiera adres URL, z którym można skontaktować się z responderem.

Wartość tego pola jest adresem URL HTTP. To pole ma priorytet w stosunku do adresu URL w rozszerzeniu certyfikatu AuthorityInfoAccess (AIA).

Wartość jest ignorowana, chyba że spełnione są oba poniższe instrukcje:

- Struktura MQAIR jest w wersji 2 lub nowszej (pole Wersja jest ustawione na wartość MQAIR_VERSION_2 lub większe).
- Pole Typ AuthInfo jest ustawione na wartość MQAIT_OCSP.

Jeśli pole nie zawiera adresu URL HTTP w poprawnym formacie (i nie jest on ignorowany), wywołanie MQCONN nie powiedzie się i zostanie zakodowany kod przyczyny MQRC_OCSP_URL_ERROR.

W tym polu jest rozróżniana wielkość liter. Musi on rozpoczynać się od łańcucha http:// w postaci małych liter. W pozostałej części adresu URL może być rozróżniana wielkość liter, w zależności od implementacji serwera OCSP.

To pole nie podlega konwersji danych.

StrucId (MQCHAR4)

Wartość musi być następująca:

MQAIR_STRUC_ID

Identyfikator rekordu informacji uwierzytelniającej.

Dla języka programowania C zdefiniowana jest również stała MQAIR_STRUC_ID_ARRAY; ma taką samą wartość jak MQAIR_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQAIR_STRUC_ID.

Wersja (MQLONG)

Numer wersji struktury MQAIR.

Wartość musi być jedną z następujących wartości:

MQAIR_VERSION_1

Rekord informacji uwierzytelniających Version-1 .

MQAIR_VERSION_2

Rekord informacji uwierzytelniających Version-2 .

Następująca stała określa numer wersji bieżącej wersji:

MQAIR_CURRENT_VERSION

Bieżąca wersja rekordu informacji uwierzytelniającej.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQAIR_VERSION_1.

Wartości początkowe i deklaracje języków dla produktu MQAIR

Tabela 13. Początkowe wartości pól w MQAIR		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQAIR_STRUC_ID	'AIR¬'
Wersja	MQAIR_VERSION_1	1
Typ AuthInfo	MQAIT_CRL_LDAP	1
AuthInfoConnName	Brak	Pusty łańcuch lub odstępy
LDAPUserNamePtr	Brak	Pusty wskaźnik lub zerowe bajty
Przesunięcie LDAPUserName	Brak	0
Długość elementu LDAPUserName	Brak	0
Hasło_LDAPPassword	Brak	Pusty łańcuch lub odstępy
OCSPResponderURL	Brak	Pusty łańcuch lub odstępy
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		
2. W języku programowania C: zmienna makraParametr MQAIR_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQAIR MyAIR = {MQAIR_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     AuthInfoType;     /* Type of authentication
                                information */
    MQCHAR264  AuthInfoConnName; /* Connection name of CRL LDAP
                                server */
    PMQCHAR    LDAPUserNamePtr;  /* Address of LDAP user name */
    MQLONG     LDAPUserNameOffset; /* Offset of LDAP user name from start
                                of MQAIR structure */
    MQLONG     LDAPUserNameLength; /* Length of LDAP user name */
}
```

```

MQCHAR32  LDAPPassword;      /* Password to access LDAP server */
MQCHAR256 OCSPResponderURL;  /* URL of OCSP responder */

};

```

Deklaracja języka COBOL

```

**  MQAIR structure
10 MQAIR.
**  Structure identifier
15 MQAIR-STRUCID          PIC X(4).
**  Structure version number
15 MQAIR-VERSION          PIC S9(9) BINARY.
**  Type of authentication information
15 MQAIR-AUTHINFOTYPE     PIC S9(9) BINARY.
**  Connection name of CRL LDAP server
15 MQAIR-AUTHINFOCONNNAME PIC X(264).
**  Address of LDAP user name
15 MQAIR-LDAPUSERNAMEPTR  POINTER.
**  Offset of LDAP user name from start of MQAIR structure
15 MQAIR-LDAPUSERNAMEOFFSET PIC S9(9) BINARY.
**  Length of LDAP user name
15 MQAIR-LDAPUSERNAMELENGTH PIC S9(9) BINARY.
**  Password to access LDAP server
15 MQAIR-LDAPPASSWORD     PIC X(32).
**  URL of OCSP responder
15 MQAIR-OCSPRESPONDERURL PIC X(256).

```

Wizualna deklaracja podstawowa

```

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-Opcje uchwytu buforu do obsługi komunikatów

W poniższej tabeli podsumowano pola w strukturze. MQBMHO struktura-bufor do opcji uchwytu komunikatu

Tabela 14. Pola w MQBMHO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem komendy MQBMHO	Opcje

Przegląd produktu MQBMHO

Dostępność: Wszystko. Struktura opcji uchwytu buforu do obsługi komunikatów-przegląd

Przeznaczenie: Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki uchwyt komunikatów są generowane z buforów. Struktura jest parametrem wejściowym w wywołaniu MQBUFMH.

Zestaw znaków i kodowanie: Dane w tabeli MQBMHO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla MQBMHO

Struktura opcji uchwytu komunikatu buforu do komunikatu-pola

Struktura MQBMHO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

Struktura uchwytu komunikatu do struktury uchwytu komunikatu-pole Opcje

Możliwe wartości:

MQBMHO_DELETE_PROPERTIES

Właściwości, które są dodawane do uchwytu komunikatu, są usuwane z buforu. Jeśli wywołanie nie powiedzie się, żadne właściwości nie zostaną usunięte.

Opcje domyślne: Jeśli nie jest potrzebna opisana opcja, należy użyć następującej opcji:

MQBMHO_NONE

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQBMHO_DELETE_PROPERTIES.

StrucId (MQCHAR4)

Struktura uchwytu buforu do komunikatu-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQBMHO_STRUC_ID

Identyfikator dla struktury uchwytu komunikatu dla buforu.

Dla języka programowania w języku C jest również zdefiniowana stała MQBMHO_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQBMHO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQBMHO_STRUC_ID.

Wersja (MQLONG)

Bufor do struktury uchwytu komunikatu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQBMHO_VERSION_1

Numer wersji dla struktury uchwytu buforu do komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQBMHO_CURRENT_VERSION

Bieżąca wersja buforu do struktury uchwytu komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQBMHO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQBMHO

Struktura uchwytu buforu do komunikatu-wartości początkowe

Tabela 15. Początkowe wartości pól w MQBMHO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQBMHO_STRUC_ID	' BMHO '
<i>Version</i>	MQBMHO_VERSION_1	1
<i>Options</i>	MQBMHO_NONE	0

Tabela 15. Początkowe wartości pól w MQBMHO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: 1. W języku programowania C: zmienna makraParametr MQBMHO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQBMHO MyBMHO = {MQBMHO_DEFAULT};</pre>		

Deklaracja C

Struktura obsługi komunikatów w buforze-deklaracja języka C

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG Version;           /* Structure version number */
    MQLONG Options;           /* Options that control the action of
                               MQBUFMH */
};
```

Deklaracja języka COBOL

Struktura obsługi komunikatów w buforze-deklaracja języka COBOL

```
** MQBMHO structure
10 MQBMHO.
** Structure identifier
15 MQBMHO-STRUCID PIC X(4).
** Structure version number
15 MQBMHO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQBUFMH
15 MQBMHO-OPTIONS PIC S9(9) BINARY.
```

Deklaracja PL/I

Struktura uchwytu buforu do struktury uchwytu komunikatu-deklaracja języka PL/I

```
Dcl
1 MQBMHO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 Options fixed bin(31), /* Options that control the action
                           of MQBUFMH */
```

Deklaracja High Level Assembler

Struktura uchwytu buforu do struktury uchwytu komunikatu-Deklaracja języka assemblera

```
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-opcje rozpoczęcia

W poniższej tabeli podsumowano pola w strukturze.

Tabela 16. Pola w obiekcie MQBO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem komendy MQBEGIN	Opcje

Przegląd dla obiektu MQBO

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows ; Nie jest dostępne dla IBM MQ MQI clients.

Cel: Struktura MQBO umożliwia aplikacji określanie opcji związanych z tworzeniem jednostki pracy. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu komendy MQBEGIN.

Zestaw znaków i kodowanie: Dane w obiekcie MQBO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla obiektu MQBO

Struktura MQBO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

To pole jest zawsze polem wejściowym. Jego początkowa wartość to MQBO_NONE.

Wartość musi być następująca:

MQBO_NONE

Nie określono żadnych opcji.

StrucId (MQCHAR4)

To pole jest zawsze polem wejściowym. Jego początkowa wartość to MQBO_STRUC_ID.

Wartość musi być następująca:

Identyfikator MQBO_STRUC_ID

Identyfikator struktury opcji begin-options.

Dla języka programowania C zdefiniowana jest również stała MQBO_STRUC_ID_ARRAY; ma taką samą wartość jak MQBO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG)

To pole jest zawsze polem wejściowym. Jego początkowa wartość to MQBO_VERSION_1.

Wartość musi być następująca:

MQBO_VERSION_1

Numer wersji dla struktury opcji begin-options.

Następująca stała określa numer wersji bieżącej wersji:

MQBO_CURRENT_VERSION

Bieżąca wersja struktury opcji begin-options.

Wartości początkowe i deklaracje języka dla obiektu MQBO

Tabela 17. Początkowe wartości pól w MQBO dla MQBO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	Identyfikator MQBO_STRUC_ID	'B0¬¬'
<i>Version</i>	MQBO_VERSION_1	1

Tabela 17. Początkowe wartości pól w MQBO dla MQBO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>Options</i>	MQBO_NONE	0
Uwagi: - Symbol ↵ reprezentuje pojedynczy pusty znak. - W języku programowania C: zmienna makraParametr MQBO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQBO MyBO = {MQBO_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
    MQCHAR4 StrucId; /* Structure identifier */
    MQLONG Version; /* Structure version number */
    MQLONG Options; /* Options that control the action of MQBEGIN */
};
```

Deklaracja języka COBOL

```
** MQBO structure
10 MQBO.
** Structure identifier
15 MQBO-STRUCID PIC X(4).
** Structure version number
15 MQBO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQBEGIN
15 MQBO-OPTIONS PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQBO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 Options fixed bin(31); /* Options that control the action of
MQBEGIN */
```

Wizualna deklaracja podstawowa

```
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-kontekst wywołania zwrotnego

W poniższej tabeli podsumowano pola w strukturze. Struktura opisująca procedurę zwrotną.

Tabela 18. Pola w tabeli MQCBC		
Pole	Opis	Temat
<i>StrucID</i>	Identyfikator struktury	StrucID

Tabela 18. Pola w tabeli MQCBC (kontynuacja)		
Pole	Opis	Temat
<i>Version</i>	Numer wersji struktury	Wersja
<i>CallType</i>	Dlaczego funkcja została wywołana	CallType
<i>Hobj</i>	Uchwyt obiektu	Hobj
<i>CallbackArea</i>	Pole dla funkcji zwrotnej do użycia	CallbackArea
<i>ConnectionArea</i>	Pole dla funkcji zwrotnej do użycia	ConnectionArea
<i>CompCode</i>	Kod zakończenia	CompCode
<i>Reason</i>	Kod przyczyny	Powód
<i>State</i>	Wskazanie stanu obecnego konsumenta	STATE
<i>DataLength</i>	Długość komunikatu	DataLength
<i>BufferLength</i>	Długość buforu komunikatów w bajtach	BufferLength
<i>Flags</i>	Flagi ogólne	Flagi
Uwaga: Pozostałe pole jest ignorowane, jeśli wersja jest mniejsza niż MQCBC_VERSION_2		
<i>ReconnectDelay</i>	Liczba milisekund przed podjęciem próby ponownego nawiązania połączenia	ReconnectDelay

Przegląd produktu MQCBC

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS, plus IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQCBC służy do określania informacji kontekstowych przekazywanych do funkcji zwrotnej. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu procedury konsumenta komunikatów.

Wersja: Bieżąca wersja tabeli MQCBC to MQCBC_VERSION_2.

Zestaw znaków i kodowanie: Dane w tabeli MQCBC muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ, struktura ta będzie znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla tabeli MQCBC

Alfabetyczna lista pól dla struktury MQCBC.

Struktura MQCBC zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

BufferLength (MQLONG)

To pole jest długością w bajtach buforu komunikatów, który został przekazany do tej funkcji.

Bufor może być większy niż wartość zarówno MaxMsg(wartość długości) zdefiniowana dla konsumenta, jak i wartość ReturnedLength (w przypadku wartości MQGMO).

Rzeczywista długość komunikatu jest podana w polu [DataLength](#).

Aplikacja może wykorzystać cały bufor do własnych celów przez czas trwania funkcji zwrotnej.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji obsługi wyjątku.

CallbackArea (MQPTR)

To pole jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian z pola *CallbackArea* w strukturze MQCBD, która jest parametrem w wywołaniu MQCB używanym do definiowania funkcji zwrotnej.

Zmiany w *CallbackArea* są zachowywane w wywołaniach funkcji zwrotnej dla *HObj*. To pole nie jest współużytkowane z funkcjami wywołania zwrotnego dla innych uchwytów.

Jest to pole wejściowe/wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CallType (MQLONG)

Pole zawierające informacje o tym, dlaczego ta funkcja została wywołana; zdefiniowane są następujące pola.

Typy wywołań dostarczania komunikatów: te typy wywołań zawierają informacje na temat komunikatu. Parametry *DataLength* i *BufferLength* są poprawne dla tych typów wywołań.

MQCBCT_MSG_REMOVED

Funkcja konsumenta komunikatów została wywołana z komunikatem, który został zniszczony w sposób destruktywny z uchwytu obiektu.

Jeśli wartością parametru *CompCode* jest MQCC_WARNING, wartość pola *Reason* to MQRC_TRUNCATED_MSG_ACCEPTED lub jeden z kodów wskazujących na problem konwersji danych.

MQCBCT_MSG_NOT_REMOVED

Funkcja konsumenta komunikatów została wywołana z komunikatem, który nie został jeszcze zniszczony w wyniku destrukcyjnego usunięcia z uchwytu obiektu. Komunikat może zostać destruktywnie usunięty z uchwytu obiektu przy użyciu *MsgToken*.

Możliwe, że komunikat nie został usunięty, ponieważ:

- Opcje MQGMO zażądały operacji przeglądania, MQGMO_BROWSE_*
- Komunikat jest większy niż dostępny bufor, a opcje MQGMO nie określają parametru MQGMO_ACCEPT_TRUNCATED_MSG.

Jeśli wartością parametru *CompCode* jest MQCC_WARNING, to wartością pola *Reason* jest MQRC_TRUNCATED_MSG_FAILED lub jeden z kodów wskazujących na problem konwersji danych.

Typy wywołań kontroli zwrotnej: te typy połączeń zawierają informacje o kontroli wywołania zwrotnego i nie zawierają szczegółów dotyczących komunikatu. Te typy wywołań są wymagane przy użyciu opcji Opcje w strukturze MQCBD.

Parametry *DataLength* i *BufferLength* nie są poprawne dla tych typów wywołań.

MQCBCT_REGISTER_CALL,

Celem tego typu wywołania jest umożliwienie wykonania pewnej początkowej konfiguracji przez funkcję zwrotną.

Funkcja zwrotna jest wywoływana natychmiast po zarejestrowaniu wywołania zwrotnego, czyli po powrocie z wywołania MQCB przy użyciu wartości dla pola *Operation* w tabeli MQOP_REGISTER.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i procedur obsługi zdarzeń.

Jeśli jest to wymagane, jest to pierwsze wywołanie funkcji zwrotnej.

Wartość w polu *Reason* to MQRC_NONE.

MQCBCT_START_CALL

Celem tego typu wywołania jest umożliwienie wykonania pewnej konfiguracji przez funkcję zwrotną po jej uruchomieniu, na przykład przywrócenie zasobów, które zostały wyczyszczone, gdy wcześniej została zatrzymana.

Funkcja zwrotna jest wywoływana, gdy połączenie jest uruchamiane przy użyciu komendy MQOP_START lub MQOP_START_WAIT.

Jeśli funkcja zwrotna jest zarejestrowana w innej funkcji wywołania zwrotnego, ten typ wywołania jest wywoływany po powrocie wywołania zwrotnego.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartość w polu *Reason* to MQRC_NONE.

MQCBCT_STOP_CALL,

Celem tego typu wywołania jest umożliwienie wykonania przez funkcję zwrotną niektórych procedur czyszczących po zatrzymaniu przez pewien czas, na przykład przy czyszczeniu dodatkowych zasobów, które zostały nabyte podczas korzystania z komunikatów.

Funkcja zwrotna jest wywoływana, gdy wywołanie MQCTL jest wysyłane za pomocą wartości pola *Operation* komendy MQOP_STOP.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartość pola *Reason* jest ustawiona w taki sposób, aby wskazywać przyczynę zatrzymania.

MQCBCT_DEREGISTER_CALL,

Celem tego typu wywołania jest umożliwienie wykonania przez funkcję zwrotną ostatniego czyszczenia na końcu procesu konsumowania. Funkcja zwrotna jest wywoływana, gdy:

- Funkcja zwrotna jest wyrejestrowana przy użyciu wywołania MQCB z MQOP_DEREGISTER.
- Kolejka jest zamknięta, co powoduje wyrejestrowywanie niejawnie. W tej instancji funkcja zwrotna jest przekazywana jako uchwyt obiektu MQHO_UNUSABLE_HOBJ.
- Wywołanie MQDISC kończy działanie-powoduje niejawnie zamknięcie, a tym samym wyrejestrowywanie. W tym przypadku połączenie nie jest natychmiast rozłączone, a każda bieżąca transakcja nie została jeszcze zatwierdzona.

Jeśli którekolwiek z tych działań zostanie wykonane w samej funkcji zwrotnej, działanie zostanie wywołane po powrocie wywołania zwrotnego.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i procedur obsługi zdarzeń.

Jeśli jest to wymagane, jest to ostatnie wywołanie funkcji zwrotnej.

Wartość pola *Reason* jest ustawiona w taki sposób, aby wskazywać przyczynę zatrzymania.

MQCBCT_EVENT_CALL,

Funkcja procedury obsługi zdarzeń

Funkcja procedury obsługi zdarzeń została wywołana bez komunikatu, gdy menedżer kolejek lub połączenie są zatrzymywane lub wyciszane.

To wywołanie może być użyte do podjęcia odpowiednich działań dla wszystkich funkcji zwrotnych.

Funkcja konsumenta komunikatów

Funkcja konsumenta komunikatów została wywołana bez komunikatu, gdy wykryty został błąd (*CompCode* = MQCC_FAILED), który jest specyficzny dla uchwytu obiektu; na przykład *Reason code* = MQRC_GET_INHIBITED.

Wartość pola *Reason* jest ustawiona w taki sposób, aby wskazywała przyczynę wywołania.

MQCBCT_MC_EVENT_CALL

Funkcja procedury obsługi zdarzeń została wywołana dla zdarzeń rozsyłania grupowego. Procedura obsługi zdarzeń jest wysyłana do zdarzeń IBM MQ Multicast zamiast zdarzeń normalnych IBM MQ.

Więcej informacji na temat wywołania MQCBCT_MC_EVENT_CALL zawiera sekcja [Zgłaszanie wyjątków rozsyłania grupowego](#).

CompCode (MQLONG)

To pole jest kodem zakończenia. Wskazuje, czy wystąpiły problemy z korzystaniem z komunikatu.

Wartość ta jest jedną z następujących wartości:

MQCC_OK

Pomyślne zakończenie

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

To jest pole wejściowe. Wartością początkową tego pola jest MQCC_OK.

ConnectionArea (MQPTR),

To pole jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian w polu ConnectionArea w strukturze MQCTLO. Jest to parametr wywołania MQCTL używanego do sterowania funkcją zwrótną.

Wszystkie zmiany wprowadzone w tym polu przez funkcje wywołania zwrótnego są zachowywane w obrębie wywołań funkcji zwrotnej. Ten obszar może być używany do przekazywania informacji, które mają być współużytkowane przez wszystkie funkcje wywołania zwrótnego. W przeciwieństwie do produktu *CallbackArea*, ten obszar jest wspólny dla wszystkich wywołań zwrótnych dla uchwytu połączenia.

Jest to pole wejściowe i wyjściowe. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

DataLength (MQLONG)

Jest to długość w bajtach danych aplikacji w komunikacie. Jeśli wartość jest równa zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Pole DataLength zawiera długość komunikatu, ale nie musi być długość danych komunikatu przekazanego konsumentowi. Może to być komunikat, że komunikat został obcięty. Użyj pola ReturnedLength w produkcie MQGMO, aby określić ilość danych, które zostały rzeczywiście przekazane konsumentowi.

Jeśli kod przyczyny wskazuje, że komunikat został obcięty, można użyć pola DataLength w celu określenia, jak duży jest rzeczywisty komunikat. Umożliwia to określenie wielkości buforu wymaganego do obsługi danych komunikatu, a następnie wywołanie wywołania MQCB w celu zaktualizowania wartości MaxMsgLength z odpowiednią wartością.

Jeśli zostanie podana opcja MQGMO_CONVERT, przekształcony komunikat może być większy niż wartość zwrócona przez DataLength. W takich przypadkach aplikacja prawdopodobnie musi wywołać wywołanie MQCB w celu zaktualizowania wartości MaxMsgLength w taki sposób, aby była większa niż wartość zwrócona przez menedżer kolejek dla DataLength.

Aby uniknąć problemów z obcinaniem komunikatów, należy podać wartość MaxMsg(MaxMsg) jako MQCBD_FULL_MSG_LENGTH. Powoduje to, że menedżer kolejek przydziela bufor dla pełnej długości komunikatu po konwersji danych. Należy jednak pamiętać, że nawet jeśli ta opcja jest określona, nadal możliwe jest, że wystarczająca ilość pamięci masowej nie jest dostępna, aby poprawnie przetworzyć żądanie. Aplikacje powinny zawsze sprawdzać zwrócony kod przyczyny. Na przykład, jeśli nie jest możliwe przydzielenie wystarczającej ilości pamięci masowej w celu przekształcenia komunikatu, komunikaty są zwracane do nieprzekształconego aplikacji.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

Flags (MQLONG)

Flagi zawierające informacje o tym konsumencie.

Zdefiniowana jest następująca opcja:

MQCBCF_READA_BUFFER_EMPTY

Ta opcja może zostać zwrócona, jeśli poprzednie wywołanie MQCLOSE przy użyciu opcji MQCO_QUIESCE nie powiodło się z kodem przyczyny MQRC_READ_AHEAD_MSGS.

Kod ten wskazał, że jest zwracany ostatni komunikat z wyprzedzeniem i że bufor jest teraz pusty. Jeśli aplikacja wysyła inne wywołanie MQCLOSE za pomocą opcji MQCO_QUIESCE), to zakończy się powodzeniem.

Należy zauważyć, że aplikacja nie ma gwarancji, że zostanie nadana komunikat z tym zestawem flag, ponieważ w buforze odczytu z wyprzedzeniem mogą być nadal komunikaty niezgodne z bieżącymi kryteriami wyboru. W tej instancji funkcja konsumenta jest wywoływana z kodem przyczyny MQRC_HOBJ_QUIESCED.

Jeśli bufor odczytu z wyprzedzeniem jest całkowicie pusty, konsument jest wywoływany z flagą MQCBCF_READA_BUFFER_EMPTY, a kodem przyczyny MQRC_HOBJ_QUIESCED_NO_MSGS.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

Hobj (MQHOBJ)

Jest to uchwyt obiektu dla wywołań konsumenta komunikatów.

W przypadku procedury obsługi zdarzeń ta wartość to MQHO_NONE.

Aplikacja może użyć tego uchwytu i znacznika komunikatu w bloku Opcje pobierania komunikatów, aby otrzymać komunikat, jeśli komunikat nie został usunięty z kolejki.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQHO_UNUSABLE_HOBJ

Przyczyna (MQLONG)

Jest to kod przyczyny kwalifikujący produkt *CompCode*.

To jest pole wejściowe. Wartością początkową tego pola jest MQRC_NONE.

Stan (MQLONG)

Wskazanie stanu bieżącego konsumenta. To pole jest najbardziej wartościowane do aplikacji, gdy do funkcji konsumenta przekazywany jest niezerowy kod przyczyny.

Tego pola można użyć, aby uprościć programowanie aplikacji, ponieważ nie ma potrzeby tworzenia kodu dla każdego kodu przyczyny.

To jest pole wejściowe. Wartość początkowa tego pola to MQCS_NONE.

Stan	Działanie menedżera kolejek	Wartość stałej
<i>MQCS_NONE</i> Ten kod przyczyny reprezentuje normalne wywołanie bez dodatkowych informacji o przyczynie.	Brak; jest to normalne działanie.	0
<i>MQCS_SUSPENDED_TEMPORARY</i> Te kody przyczyny reprezentują warunki tymczasowe.	Procedura zwrotna jest wywoływana w celu zgłoszenia warunku, a następnie zawieszenia. Po upływie określonego czasu system może ponowić próbę wykonania operacji, co może spowodować ponowne zwiększenie tego samego warunku.	1

Stan	Działanie menedżera kolejek	Wartość stałej
<i>MQCS_SUSPENDED_USER_ACTION</i> Te kody przyczyny reprezentują warunki, w których wywołanie zwrotne musi podjąć działanie w celu rozwiązania warunku.	Konsument jest zawieszony, a procedura zwrotna jest wywoływana w celu zgłoszenia warunku. Procedura zwrotna powinna rozstrzygać warunek, jeśli jest to możliwe, a także RESUME lub zamknąć połączenie.	2
<i>MQCS_SUSPENDED</i> Te kody przyczyny reprezentują niepowodzenia, które uniemożliwiają dalsze wywołania zwrotne komunikatu.	Menedżer kolejek automatycznie zawiesza funkcję zwrotną. Jeśli funkcja zwrotna została wznowiona, prawdopodobnie ponownie otrzyma ten sam kod przyczyny.	3
<i>MQCS_STOPPED</i> Te kody przyczyny reprezentują koniec konsumpcji komunikatów.	Dostarczono do procedury obsługi wyjątku i do wywołań zwrotnych, które określono MQCBDO_STOP_CALL. Dalsze komunikaty nie mogą być używane.	4

StrucId (MQCHAR4)

Wartość w tym polu jest identyfikatorem struktury.

Wartość musi być następująca:

MQCBC_STRUC_ID

Identyfikator struktury kontekstu wywołania zwrotnego.

W przypadku języka programowania C zdefiniowana jest również stała MQCBC_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQCBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCBC_STRUC_ID.

Wersja (MQLONG)

Wartość w tym polu jest numerem wersji struktury.

Wartość musi być następująca:

MQCBC_VERSION_1

Struktura kontekstu wywołania zwrotnego Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQCBC_CURRENT_VERSION

Bieżąca wersja struktury kontekstu wywołania zwrotnego.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCBC_VERSION_1.

Funkcja zwrotna jest zawsze przekazywana najnowsza wersja struktury.

ReconnectDelay (MQLONG)

Wartość ReconnectDelay wskazuje, jak długo menedżer kolejek będzie czekał przed podjęciem próby ponownego nawiązania połączenia. Pole może być modyfikowane przez procedurę obsługi zdarzeń w celu zmiany opóźnienia lub zatrzymania ponownego połączenia.

Użyj pola ReconnectDelay tylko wtedy, gdy wartością pola Przyczyna w kontekście wywołania zwrotnego jest MQRC_RECONNECTING.

W przypadku wpisu do procedury obsługi zdarzeń wartość ReconnectDelay to liczba milisekund, przez które menedżer kolejek będzie oczekiwać przed podjęciem próby ponownego nawiązania połączenia.

Tabela 19 na stronie 290 zawiera listę wartości, które można ustawić w celu zmodyfikowania zachowania menedżera kolejek po powrocie z procedury obsługi zdarzeń.

Tabela 19. Wartości ReconnectDelay

Nazwa	Wartość	Opis
MQRD_NO_RECONNECT	-1	Nie podejmuje się dalszych prób ponownego połączenia. Do aplikacji zwracany jest błąd.
MQRD_NO_DELAY	0	Spróbuj ponownie nawiązać połączenie natychmiast.
Milliseconds	>0	Poczekaj na tę liczbę milisekund przed ponowną próbą nawiązania połączenia.

Wartości początkowe i deklaracje języków dla MQCBC

Struktura kontekstu wywołania zwrotnego-wartości początkowe

Nie ma początkowych wartości dla struktury **MQCBC**. Struktura jest przekazywana jako parametr do procedury zwrotnej. Menedżer kolejek inicjuje strukturę; aplikacje nigdy go nie inicjują.

Deklaracja C

Struktura kontekstu wywołania zwrotnego-deklaracja języka C

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     CallType;         /* Why Function was called */
    MQHOBJ     Hobj;            /* Object Handle */
    MQPTR      CallbackArea;     /* Callback data passed to the function */
    MQPTR      ConnectionArea;   /* MQCTL data area passed to the function */
    MQLONG     CompCode;         /* Completion Code */
    MQLONG     Reason;          /* Reason Code */
    MQLONG     State;           /* Consumer State */
    MQLONG     DataLength;       /* Message Data Length */
    MQLONG     BufferLength;     /* Buffer Length */
    MQLONG     Flags;           /* Flags containing information about
                                this consumer */

    /* Ver:1 */
    MQLONG     ReconnectDelay;   /* Number of milliseconds before */
    /* Ver:2 */ };              /* reconnect attempt */
```

Deklaracja języka COBOL

```
** MQCBC structure
10 MQCBC.
** Structure Identifier
15 MQCBC-STRUCID                PIC X(4).
** Structure Version
15 MQCBC-VERSION                PIC S9(9) BINARY.
** Call Type
15 MQCBC-CALLTYPE                PIC S9(9) BINARY.
** Object Handle
15 MQCBC-HOBJ                    PIC S9(9) BINARY.
** Callback User Area
15 MQCBC-CALLBACKAREA            POINTER
** Connection Area
15 MQCBC-CONNECTIONAREA          POINTER
** Completion Code
15 MQCBC-COMPCODE                PIC S9(9) BINARY.
** Reason Code
15 MQCBC-REASON                  PIC S9(9) BINARY.
** Consumer State
15 MQCBC-STATE                   PIC S9(9) BINARY.
** Data Length
15 MQCBC-DATALENGTH              PIC S9(9) BINARY.
** Buffer Length
15 MQCBC-BUFFERLENGTH            PIC S9(9) BINARY.
** Flags
15 MQCBC-FLAGS                   PIC S9(9) BINARY.
** Ver:1 **
** Number of milliseconds before reconnect attempt
```

```

15 MQCBC-RECONNECTDELAY PIC S9(9) BINARY.
** Ver:2 **

```

Deklaracja PL/I

```

dcl
  1 MQCBC based,
  3 StrucId          char(4),          /* Structure identifier */
  3 Version          fixed bin(31),    /* Structure version */
  3 CallType         fixed bin(31),    /* Callback type */
  3 Hobj            fixed bin(31),    /* Object Handle */
  3 CallbackArea     pointer,          /* User area passed to the function */
  3 ConnectionArea   pointer,          /* Connection User Area */
  3 CompCode         fixed bin(31);    /* Completion Code */
  3 Reason           fixed bin(31);    /* Reason Code */
  3 State            fixed bin(31);    /* Consumer State */
  3 DataLength       fixed bin(31);    /* Message Data Length */
  3 BufferLength     fixed bin(31);    /* Message Buffer length */
  3 Flags            fixed bin(31);    /* Consumer Flags */
/* Ver:1 */
  3 ReconnectDelay   fixed bin(31);    /* Number of milliseconds before */
/* Ver:2 */                               /* reconnect attempt */

```

Deklaracja High Level Assembler

```

MQCBC          DSECT
MQCBC          DS 0F      Force fullword alignment
MQCBC_STRUCID  DS CL4     Structure identifier
MQCBC_VERSION  DS F       Structure version number
MQCBC_CALLTYPE DS F       Why Function was called
MQCBC_HOBJ     DS F       Object Handle
MQCBC_CALLBACKAREA DS A    Callback data passed to the function
MQCBC_CONNECTIONAREA DS A  MQCTL Data area passed to the function
MQCBC_COMPCODE DS F       Completion Code
MQCBC_REASON   DS F       Reason Code
MQCBC_STATE    DS F       Consumer State
MQCBC_DATALENGTH DS F     Message Data Length
MQCBC_BUFFERLENGTH DS F   Buffer Length
MQCBC_FLAGS    DS F       Flags containing information about this consumer
MQCBC_RECONNECTDELAY DS F  Number of milliseconds before reconnect
MQCBC_LENGTH   EQU *-MQCBC
                ORG      MQCBC
MQCBC_AREA     DS CL(MQCBC_LENGTH)

```

MQCBD-deskryptor wywołania zwrotnego

W poniższej tabeli podsumowano pola w strukturze. Struktura określająca funkcję zwrotną.

Tabela 20. Pola w tabeli MQCBD		
Pole	Opis	Temat
<i>StrucID</i>	Identyfikator struktury	StrucID
<i>Version</i>	Numer wersji struktury	Wersja
<i>CallbackType</i>	Typ funkcji zwrotnej	CallbackType
<i>Options</i>	Opcje sterujące zużyciem komunikatów	Opcje
<i>Callback Area</i>	Pole dla funkcji zwrotnej do użycia	CallbackArea
<i>CallbackFunction</i>	Określa, czy funkcja jest wywoływana jako wywołanie funkcji API	CallbackFunction
<i>CallbackName</i>	Określa, czy funkcja jest wywoływana jako program dowiązany dynamicznie	CallbackName

Tabela 20. Pola w tabeli MQCBD (kontynuacja)		
Pole	Opis	Temat
<i>MaxMsgLength</i>	Długość najdłuższego komunikatu, który może zostać odczytany	<u>MaxMsgDługość</u>

Przegląd produktu MQCBD

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSi IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQCBD służy do określania funkcji zwrotnej i opcji sterujących jej używaniem przez menedżer kolejek.

Struktura jest parametrem wejściowym w wywołaniu MQCB.

Wersja: Bieżąca wersja tabeli MQCBD ma wartość MQCBD_VERSION_1.

Zestaw znaków i kodowanie: Dane w tabeli MQCBD muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla tabeli MQCBD

Alfabetyczna lista pól dla struktury MQCBD.

Struktura MQCBD zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

CallbackArea (MQPTR)

Struktura deskryptora wywołania zwrotnego-pole CallbackArea

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian z pola CallbackArea w strukturze MQCBC, która jest parametrem w deklaracji funkcji wywołania zwrotnego.

Ta wartość jest używana tylko w przypadku *Operation* o wartości MQOP_REGISTER, która nie ma obecnie zdefiniowanego wywołania zwrotnego, nie zastępuje poprzedniej definicji.

Jest to pole wejściowe i wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CallbackFunction (MQPTR)

Struktura deskryptora wywołania zwrotnego-pole CallbackFunction

Funkcja zwrotna jest wywoływana jako wywołanie funkcji.

Użyj tego pola, aby określić wskaźnik do funkcji zwrotnej.

Użytkownik *musi* podać wartość *CallbackFunction* lub *CallbackName*. Jeśli zostaną podane obie wartości, zostanie zwrócony kod przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Jeśli nie zostanie ustawiony ani *CallbackName*, ani *CallbackFunction*, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Ta opcja nie jest obsługiwana w następującym środowisku: języki programowania i kompilatory, które nie obsługują odwołań do funkcji wskaźnika-wskaźnik. W takich sytuacjach wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

W systemie z/OS należy oczekiwać, że funkcja będzie wywoływana z konwencjami systemu operacyjnego. Na przykład w języku programowania C należy określić:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

Uwaga: W przypadku korzystania z produktu CICS z produktem IBM MQ V7.0.1 asynchroniczna konsumpcja jest obsługiwana, jeśli:

- Apar PK66866 jest stosowany do CICS TS 3.2
- Apar PK89844 jest stosowany do produktu CICS TS 4.1

CallbackName (MQCHAR128)

Struktura deskryptora wywołania zwrotnego-pole *CallbackName*

Funkcja zwrotna jest wywoływana jako dynamicznie połączony program.

Użytkownik *musi* podać wartość *CallbackFunction* lub *CallbackName*. Jeśli zostaną podane obie wartości, zostanie zwrócony kod przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Jeśli wartość *CallbackName* ani *CallbackFunction* nie jest ustawiona, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Moduł jest ładowany, gdy pierwsza procedura zwrotna do użycia jest zarejestrowana i rozładowana, gdy ostatnia procedura zwrotna ma używać jej deregisterów.

Z wyjątkiem sytuacji, w których w poniższym tekście zaznaczono, że nazwa jest wyrównana do lewej strony, bez odstępów wewnętrznych; sama nazwa jest dopełniona spacjami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

IBM i

Nazwa wywołania zwrotnego może mieć jeden z następujących formatów:

- Biblioteka "/" Program
- Library "/" ServiceProgram ("FunctionName")

Na przykład: MyLibrary/MyProgram(MyFunction).

Nazwą biblioteki może być *LIBL. Nazwy bibliotek i programów są ograniczone do maksymalnie 10 znaków.

Systemy UNIX

Nazwa wywołania zwrotnego to nazwa modułu lub biblioteki ładowanego dynamicznie, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[path]library(function)
```

Jeśli ścieżka nie jest określona, używana jest ścieżka wyszukiwania systemu.

Długość nazwy jest ograniczona do 128 znaków.

Windows

Nazwa wywołania zwrotnego jest nazwą biblioteki dołączanej dynamicznie, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu i napędem:

```
[d:][path]library(function)
```

Jeśli napęd i ścieżka nie są określone, używana jest ścieżka wyszukiwania systemu.

Długość nazwy jest ograniczona do 128 znaków.

z/OS

Nazwa wywołania zwrotnego to nazwa modułu ładowalnego, który jest poprawny dla specyfikacji w parametrze EP makra LINK lub LOAD.

Długość nazwy jest ograniczona do 8 znaków.

z/OS CICS

Nazwa wywołania zwrotnego to nazwa modułu ładowalnego, który jest poprawny dla specyfikacji w parametrze PROGRAM makra komendy EXEC CICS LINK.

Długość nazwy jest ograniczona do 8 znaków.

Program może być zdefiniowany jako zdalny przy użyciu opcji REMOTESYTEM zainstalowanej definicji programu lub przez dynamiczny program routingu.

Zdalny region CICS musi być połączony z programem IBM MQ, jeśli program ma używać wywołań funkcji API IBM MQ. Należy jednak pamiętać, że pole Hobj w strukturze MQCBC nie jest poprawne w systemie zdalnym.

Jeśli wystąpi błąd podczas próby załadowania programu *CallbackName*, do aplikacji zwracany jest jeden z następujących kodów błędów:

- MQRC_MODULE_NOT_FOUND
- Niepoprawna wartość MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

W dzienniku błędów zapisywany jest również komunikat zawierający nazwę modułu, dla którego podjęto próbę ładowania, oraz kod przyczyny niepowodzenia z systemu operacyjnego.

To jest pole wejściowe. Początkowa wartość tego pola jest łańcuchem pustym lub pustym.

CallbackType (MQLONG)

Struktura deskryptora wywołania zwrotnego-pole CallbackType

Jest to typ funkcji zwrotnej. Wartość musi być jedną z następujących wartości:

MQCBT_MESSAGE_CONSUMER

Definiuje tę procedurę zwrotną jako funkcję konsumenta komunikatów.

Funkcja zwrotna konsumenta komunikatu jest wywoływana, gdy komunikat, spełniający określone kryteria wyboru, jest dostępny na uchwycie obiektu i połączenie jest uruchamiane.

MQCBT_EVENT_HANDLER

Definiuje tę procedurę zwrotną jako asynchroniczną procedurę zdarzeń. Nie jest ona kierowana do odbierania komunikatów dla uchwytu.

Program *Hobj* nie jest wymagany w wywołaniu obiektu MQCB definiującym procedurę obsługi zdarzeń i jest ignorowany, jeśli jest określony.

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko konsumenta komunikatów. Funkcja konsumenta jest wywoływana bez komunikatu, gdy wystąpi zdarzenie, na przykład menedżer kolejek lub połączenie zatrzymujące się lub wygaszające. Nie jest wywoływana dla warunków, które są specyficzne dla pojedynczego konsumenta komunikatów, na przykład MQRC_GET_INHIBITED.

Zdarzenia są dostarczane do aplikacji, niezależnie od tego, czy połączenie zostało uruchomione, czy zatrzymane, z wyjątkiem następujących środowisk:

- CICS w środowisku z/OS
- aplikacje wielowątkowe

Jeśli program wywołujący nie przekaże jednej z tych wartości, wywołanie zakończy się niepowodzeniem z kodem *Reason* o wartości MQRC_CALLBACK_TYPE_ERROR.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQCBT_MESSAGE_CONSUMER.

MaxMsgDługość (MQLONG)

Jest to długość (w bajtach) najdłuższego komunikatu, który może zostać odczytany z uchwytu i podany do procedury zwrotnej. Struktura deskryptora wywołania zwrotnego-pole długości MaxMsg

Jeśli komunikat ma dłuższą długość, procedura zwrotna otrzymuje *MaxMsgLength* bajtów komunikatu, a kod przyczyny:

- MQRC_TRUNCATED_MSG_FAILED lub
- MQRC_TRUNCATED_MSG_ACCEPTED (jeśli określono wartość MQGMO_ACCEPT_TRUNCATED_MSG).

Rzeczywista długość komunikatu jest podana w polu DataLength w strukturze MQCBC.

Zdefiniowane są następujące wartości specjalne:

MQCBD_FULL_MSG_LENGTH

Długość buforu jest korygowana przez system w taki sposób, aby komunikaty były zwracane bez obcinania.

Jeśli dostępna jest niewystarczająca ilość pamięci, aby przydzielić bufor do odebrania komunikatu, system wywołuje funkcję zwrotną z kodem przyczyny MQRC_STORAGE_NOT_AVAILABLE.

Jeśli na przykład zostanie wysłane żądanie konwersji danych, a ilość pamięci jest niewystarczająca do przekształcenia danych komunikatu, wówczas nieprzekształcony komunikat jest przekazywany do funkcji zwrotnej.

To jest pole wejściowe. Początkowa wartość pola *MaxMsgLength* to MQCBD_FULL_MSG_LENGTH.

Opcje (MQLONG)

Struktura deskryptora wywołania zwrotnego-pole Opcje

Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

MQCBDO_FAIL_IF QUIESCING

Wywołanie MQCB nie powiedzie się, jeśli menedżer kolejek znajduje się w stanie wygaszania.

W systemie z/OSa opcja wymusza również, aby wywołanie MQCB nie powiodło się, jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wygaszania.

Podaj wartość MQGMO_FAIL_IF QUIESCING, w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadomienie konsumentów komunikatów, gdy są one wygaszane.

Opcje sterujące: Następujące opcje kontrolują, czy funkcja zwrotna jest wywoływana, bez komunikatu, kiedy stan zmienia się w konsumencie:

Wywołanie MQCBDO_REGISTER_CALL

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT_REGISTER_CALL.

Wywołania MQCBDO_START_CALL

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT_START_CALL.

Wywołanie MQCBDO_STOP_CALL

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT_STOP_CALL.

Wywołanie MQCBDO_DEREGISTER_CALL

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT_DEREGISTER_CALL.

MQCBDO_EVENT_CALL,

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENT_CALL

Funkcja zwrotna jest wywoływana z wywołaniem typu MQCBCT__MC_EVENT_CALL.

Więcej informacji na temat tych typów wywołań znajduje się w sekcji CallType.

Opcja domyślna: Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

MQCBDO_NONE

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

Parametr MQCBDO_NONE jest zdefiniowany w celu uzyskania dokumentacji programu pomocowego. Nie jest on przeznaczony do użycia z żadną inną opcją, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

To jest pole wejściowe. Wartością początkową pola *Options* jest MQCBDO_NONE.

StrucId (MQCHAR4)

Struktura deskryptora wywołania zwrotnego-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQCBD_STRUC_ID

Identyfikator struktury deskryptora wywołania zwrotnego.

Dla języka programowania C zdefiniowana jest również stała MQCBD_STRUC_ID_ARRAY. Ma ona taką samą wartość co identyfikator MQCBD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQCBD_STRUC_ID.

Wersja (MQLONG)

Struktura deskryptora wywołania zwrotnego-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQCBD_VERSION_1

Struktura deskryptora wywołania zwrotnego Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQCBD_CURRENT_VERSION

Bieżąca wersja struktury deskryptora wywołania zwrotnego.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCBD_VERSION_1.

Wartości początkowe i deklaracje języków dla MQCBD

Struktura deskryptora wywołania zwrotnego-wartości początkowe

Tabela 21. Początkowe wartości pól w MQCBD		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQCBD_STRUC_ID	' CBD↵ '
<i>Version</i>	MQCBD_VERSION_1	1
<i>CallBackType</i>	MQCBT_MESSAGE_CONSUMER	1
<i>Options</i>	MQCBDO_NONE	0
<i>CallbackArea</i>	Brak	Pusty wskaźnik lub puste odstępy
<i>CallbackFunction</i>	Brak	Pusty wskaźnik lub puste odstępy
<i>CallbackName</i>	Brak	Pusty łańcuch lub odstępy
<i>MaxMsgLength</i>	MQCBD_FULL_MSG_LENGTH	-1

Tabela 21. Początkowe wartości pól w MQCBD (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: - Symbol – reprezentuje pojedynczy pusty znak. - Wartość pusta lub wartość pusta oznacza wartość null w języku programowania C, a puste znaki w innych językach programowania. - W języku programowania C: zmienna makraMQCBD_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQCBD MyCBD = {MQCBD_DEFAULT};</pre>		

Deklaracja C

Struktura deskryptora wywołania zwrotnego-deklaracja języka C

```
typedef struct tagMQCBD MQCBD;
struct tagMQCBD {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQLONG       CallbackType;     /* Callback function type */
    MQLONG       Options;          /* Options controlling message
                                   consumption */
    MQPTR        CallbackArea;     /* User data passed to the function */
    MQPTR        CallbackFunction; /* Callback function pointer */
    MQCHAR128    CallbackName;     /* Callback name */
    MQLONG       MaxMsgLength;     /* Maximum message length */
};
```

Deklaracja języka COBOL

```
** MQCBCD structure
10  MQCBD.
** Structure Identifier
15  MQCBD-STRUCID                PIC X(4).
** Structure Version
15  MQCBD-VERSION               PIC S9(9) BINARY.
** Callback Type
15  MQCBD-CALLBACKTYPE          PIC S9(9) BINARY.
** Options
15  MQCBD-OPTIONS               PIC S9(9) BINARY.
** Callback User Area
15  MQCBD-CALLBACKAREA          POINTER
** Callback Function Pointer
15  MQCBD-CALLBACKFUNCTION       FUNCTION-POINTER
** Callback Program Name
15  MQCBD-CALLBACKNAME           PIC X(128)
** Maximum Message Length
15  MQCDB-MAXMSGLENGTH          PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1  MQCBD based,
3  StrucId          char(4),          /* Structure identifier*/
3  Version          fixed bin(31),    /* Structure version*/
3  CallbackType     fixed bin(31),    /* Callback function type */
3  Options          fixed bin(31),    /* Options */
3  CallbackArea     pointer,          /* User area passed to the function */
3  CallbackFunction pointer,          /* Callback Function Pointer */
3  CallbackName     char(128),        /* Callback Program Name */
3  MaxMsgLength     fixed bin(31);    /* Maximum Message Length */
```

MQCHARV-łańcuch o zmiennej długości

W poniższej tabeli podsumowano pola w strukturze.

Pole	Opis	Temat
<i>VSPtr</i>	Wskaźnik do łańcucha o zmiennej długości	VSPtr
<i>VSOffset</i>	Przesunięcie w bajtach zmiennej długości zmiennej długości od początku struktury zawierającej tę strukturę MQCHARV	Zestaw VSOffset
<i>VSLength</i>	Długość (w bajtach) łańcucha zmiennej długości adresowana przez pole VSPtr lub VSOffset.	Długość fali
<i>VSBufSize</i>	Wielkość buforu zajętego przez pole VSPtr lub VSOffset w bajtach.	VSBufSize
<i>VSCCSID</i>	Identyfikator zestawu znaków dla łańcucha o zmiennej długości, który jest adresowany przez pole VSPtr lub VSOffset.	Identyfikator VSCCSID

Przegląd produktu MQCHARV

Dostępność: AIX, HP-UX, Solaris, Linux, IBM i, Windows oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Użyj struktury MQCHARV, aby opisać łańcuch o zmiennej długości.

Zestaw znaków i kodowanie: Dane w tabeli MQCHARV muszą znajdować się w kodowaniu menedżera kolejek lokalnych, który jest nadawany przez komendę MQENC_NATIVE i zestaw znaków pola VSCCSID w strukturze. Jeśli aplikacja jest uruchomiona jako klient MQ, struktura musi znajdować się w kodowaniu klienta. Niektóre zestawy znaków mają reprezentację, która zależy od kodowania. Jeśli parametr VSCCSID jest jednym z tych zestawów znaków, używane kodowanie jest takie samo, jak kodowanie innych pól w tabeli MQCHARV. Zestaw znaków identyfikowany przez VSCCSID może być zestawem znaków dwubajtowych (DBCS).

Użycie: Struktura MQCHARV odnosi się do danych, które mogą być nieciągłe ze strukturą zawierającą ją. W celu rozwiązania tych danych można użyć pól zadeklarowanych z typem danych wskaźnika. Należy pamiętać, że język COBOL nie obsługuje wskaźnika typu danych we wszystkich środowiskach. Z tego powodu dane mogą być również adresowane za pomocą pól, które zawierają przesunięcie danych od początku struktury zawierającej MQCHARV.

programowanie w języku COBOL

Aby port był używany między środowiskami, należy upewnić się, że typ danych wskaźnika jest dostępny we wszystkich planowanych środowiskach. Jeśli nie, aplikacja musi zwracać się do danych, korzystając z pól przesunięcia zamiast pól wskaźnika.

W tych środowiskach, w których nie są obsługiwane wskaźniki, można zadeklarować pola wskaźnika jako łańcuchy bajtowe o odpowiedniej długości, przy czym wartość początkowa to łańcuch bajtowy o wartości all-null. Nie należy zmieniać tej wartości początkowej, jeśli używane są zmienne przesunięcia. Jednym ze sposobów, aby to zrobić bez zmiany dostarczonych ksiąg kopii, jest użycie następujących elementów:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

gdzie CMQCHRVV może być wymieniany na potrzeby książki kopiowej.

Pola dla tabeli MQCHARV

Struktura MQCHARV zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**.

VSBufSize (MQLONG)

Jest to wielkość w bajtach buforu, do której adresowane są pola VSPtr lub VSOOffset.

Jeśli struktura MQCHARV jest używana jako pole wyjściowe w wywołaniu funkcji, to pole musi być inicjowane z długością udostępnionego buforu. Jeśli wartość parametru VSLength jest większa niż VSBufSize, do programu wywołującego w buforze zwracane są tylko bajty danych VSBufSize.

Ta wartość musi być wartością większą lub równą zero lub następującą wartością specjalną, która jest rozpoznawana:

MQVS_USE_VSLENGTH

Jeśli zostanie podana, długość buforu jest pobierana z pola długości VSLength w strukturze MQCHARV. Nie należy używać tej wartości w przypadku używania struktury jako pola wyjściowego, a bufor jest udostępniany.

Jest to początkowa wartość tego pola.

VSCCSID (MQLONG)

Jest to identyfikator zestawu znaków dla łańcucha o zmiennej długości, który jest adresowany w polu **VSPtr** lub **VSOOffset**.

Wartością początkową tego pola jest *MQCCSI_APPL*, która jest zdefiniowana przez program MQ w celu wskazania, że powinna zostać zmieniona na prawdziwy identyfikator zestawu znaków dla bieżącego procesu. W wyniku tego wartość stałej *MQCCSI_APPL* nigdy nie jest powiązana z łańcuchem o zmiennej długości.

Początkową wartość tego pola można zmienić, definiując inną wartość dla stałej *MQCCSI_APPL* dla danej jednostki kompilacji. To, jak to zrobić zależy od języka programowania aplikacji.

 W systemach z/OS domyślna aplikacja CCSID używana przez *MQCCSI_APPL* jest zdefiniowana w następujący sposób:

- W przypadku aplikacji zadania wsadowego LE za pomocą interfejsu DLL wartością domyślną jest CODESET powiązana z bieżącymi ustawieniami narodowymi w momencie wydawania **MQCONN** (wartość domyślna to 1047).
- W przypadku aplikacji typu LE powiązanych z jednym z wsadowych kodów pośredniczących produktu MQ wartością domyślną jest CODESET powiązana z bieżącymi ustawieniami narodowymi w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku zadań wsadowych innych niż LE działających w wątku USS, wartością domyślną jest wartość THLICCSID w momencie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku innych aplikacji wsadowych domyślnie jest to CCSID menedżera kolejek.

Długość VSLength (MQLONG)

Długość (w bajtach) łańcucha zmiennej długości adresowana przez pole VSPtr lub VSOOffset.

Wartością początkową tego pola jest 0. Wartość musi być większa lub równa zero lub następująca wartość specjalna jest rozpoznawana:

MQVS_NULL_TERMINATED,

Jeśli parametr MQVS_NULL_TERMINATED nie zostanie określony, bajty VSLength są dołączane jako część łańcucha. Jeśli występują znaki o wartości NULL, nie są one ograniczane przez łańcuch.

Jeśli określono wartość MQVS_NULL_TERMINATED, łańcuch jest ograniczany przez pierwsze wartości null napotkane w łańcuchu. Sama wartość NULL nie jest uwzględniana jako część tego łańcucha.

Uwaga: Znak o kodzie zero używany do zakończenia łańcucha, jeśli określona jest wartość MQVS_NULL_TERMINATED, to wartość NULL z zestawu kodowego określonego przez VSCCSID.

Na przykład w kodowaniu UTF-16 (CCSIDUCS-2 1200 i 13488) jest to dwubajtowe kodowanie Unicode, w którym wartość null jest reprezentowana przez 16-bitową liczbę wszystkich zer. W UTF-16 często spotykano pojedyncze bajty ustawione na wszystkie zero, które są częścią znaków (na przykład 7-bitowe znaki ASCII), ale łańcuchy będą zakończone znakiem NULL tylko wtedy, gdy na granicy

parzystej bajtu zostaną znalezione dwa bajty "zero". Możliwe jest uzyskanie dwóch "zerowych" bajtów na granicy nieparzystej, gdy są one każdą częścią poprawnych znaków. Na przykład x'01' x'00 x'00 'x' 30 ' reprezentuje dwa poprawne znaki Unicode i nie ma wartości null w przypadku zakończenia łańcucha.

VOffset (MQLONG)

Przesunięcie może być dodatnie lub ujemne. Można użyć pola VSPtr lub VSOOffset, aby określić łańcuch o zmiennej długości, ale nie oba te łańcuchy. Przesunięcie (w bajtach) łańcucha o zmiennej długości od początku tabeli MQCHARV lub struktury zawierającej ją.

Jeśli struktura MQCHARV jest osadzona w innej strukturze, ta wartość jest przesuniętą w bajtach zmiennej długości łańcucha od początku struktury zawierającej tę strukturę MQCHARV. Jeśli struktura MQCHARV nie jest osadzona w innej strukturze, na przykład jeśli jest ona określona jako parametr w wywołaniu funkcji, to przesunięcie jest względne wobec początku struktury MQCHARV.

Wartością początkową tego pola jest 0.

VSPtr (MQPTR)

Jest to wskaźnik do łańcucha o zmiennej długości.

Można użyć pola VSPtr lub VSOOffset, aby określić łańcuch o zmiennej długości, ale nie oba te łańcuchy.

Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

Wartości początkowe i deklaracje języków dla tabeli MQCHARV

Początkowe wartości pól w tabeli MQCHARV

Nazwa pola	Nazwa stałej	Wartość stałej
<i>VSPtr</i>	Brak	Pusty wskaźnik lub zerowa liczba bajtów.
<i>VSOOffset</i>	Brak	0
<i>VBufSize</i>	MQVS_USE_VSLENGTH	0
<i>VSLength</i>	Brak	0
<i>VSCCSID</i>	MQCCSI_APPL	-3
Uwaga: W języku programowania C zmienna makra MQCHARV_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQCHARV MyVarStr = {MQCHARV_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQCHARV MQCHARV;
struct tagMQCHARV {
    MQPTR      VSPtr;           /* Address of variable length string */
    MQLONG     VSOOffset;       /* Offset of variable length string */
    MQLONG     VBufSize;        /* Size of buffer */
    MQLONG     VSLength;        /* Length of variable length string */
    MQLONG     VSCCSID;         /* CCSID of variable length string */
};
```

Deklaracja języka COBOL dla MQCHARV

```
** MQCHARV structure
10 MQCHARV.
```



```

** Address of variable length string
15 MQCHARV-VSPTR      POINTER.
** Offset of variable length string
15 MQCHARV-VSOFFSET    PIC S9(9) BINARY.
** Size of buffer
15 MQCHARV-VSBUFSIZE   PIC S9(9) BINARY.
** Length of variable length string
15 MQCHARV-VSLENGTH    PIC S9(9) BINARY.
** CCSID of variable length string
15 MQCHARV-VSCCSID     PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQCHARV based,
3 VSPtr      pointer,      /* Address of variable length string */
3 VSOffset    fixed bin(31), /* Offset of variable length string */
3 VSBufSize   fixed bin(31), /* Size of buffer */
3 VSLength    fixed bin(31), /* Length of variable length string */
3 VSCCSID     fixed bin(31); /* CCSID of variable length string */

```

Deklaracja High Level Assembler

```

MQCHARV          DSECT
MQCHARV_VSPTR     DS   F      Address of variable length string
MQCHARV_VSOFFSET  DS   F      Offset of variable length string
MQCHARV_VSBUFSIZE DS   F      Size of buffer
MQCHARV_VSLENGTH  DS   F      Length of variable length string
MQCHARV_VSCCSID   DS   F      CCSID of variable length string
*
MQCHARV_LENGTH    EQU  *-MQCHARV
ORG  MQCHARV
MQCHARV_AREA      DS      CL(MQCHARV_LENGTH)

```

Redefinicja MQCCSI_APPL

W poniższych przykładach przedstawiono sposób nadpisania wartości parametru MQCCSI_APPL w różnych językach programowania. Można zmienić wartość parametru MQCCSI_APPL, usuwając konieczność ustawienia wartości VSCCSID dla każdego łańcucha długości zmiennej oddzielnie.

W tych przykładach identyfikator CCSID jest ustawiony na wartość 1208; należy zmienić tę wartość na wartość, która jest wymagana. Staje się to wartością domyślną, którą można przesłonić, ustawiając wartość VSCCSID w dowolnej konkretnej instancji MQCHARV.

Użycie C

```

#define MQCCSI_APPL 1208
#include <cmqc.h>

```

Składnia języka COBOL

```

COPY CMQXYZV REPLACING -3 BY 1208.

```

Użycie języka PL/I

```

%MQCCSI_APPL = '1208';
%include syslib(cmqp);

```

Użycie asemblera System/390

MQCCSI_APPL EQU 1208
CMQA LIST=NO

MQCIH-nagłówek CICS bridge

W poniższej tabeli podsumowano pola w strukturze.

Tabela 22. Pola w tabeli MQCIH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQCIH	StrucLength
<i>Encoding</i>	Zarezerwowane	Kodowanie
<i>CodedCharSetId</i>	Zarezerwowane	CodedCharSetId
<i>Format</i>	Nazwa formatu produktu MQ , która jest zgodna z nazwą MQCIH	Formatowanie
<i>Flags</i>	Flagi	Flagi
<i>ReturnCode</i>	Kod powrotu z mostu	ReturnCode
<i>CompCode</i>	Kod zakończenia MQ lub CICS EIBRESP	CompCode
<i>Reason</i>	Kod przyczyny lub opinii MQ lub CICS EIBRESP2	Powód
<i>UOWControl</i>	Element sterujący jednostki pracy	UOWControl
<i>GetWaitInterval</i>	Odstęp czasu oczekiwania dla wywołania MQGET wystawionego przez zadanie mostu	GetWaitInterwał
<i>LinkType</i>	Typ odsyłacza	LinkType
<i>OutputDataLength</i>	Długość danych wyjściowych komendy COMMAREA	OutputDataDługość
<i>FacilityKeepTime</i>	Czas zwolnienia obiektu pomostowego	FacilityKeepCzas
<i>ADSDescriptor</i>	Wysyłaj/odbieraj deskryptor ADS	Deskryptor ADSDescriptor
<i>ConversationalTask</i>	Określa, czy zadanie może być konwersacyjne	ConversationalTask
<i>TaskEndStatus</i>	Status na końcu zadania	TaskEndStatus
<i>Facility</i>	Znacznik obiektu pomostowego	Udogodnienia
<i>Function</i>	Nazwa wywołania MQ lub funkcja CICS EIBFN	Function
<i>AbendCode</i>	Kod abend	AbendCode
<i>Authenticator</i>	Hasło lub passticket	Authenticator
<i>Reserved1</i>	Zarezerwowane	Reserved1
<i>ReplyToFormat</i>	Nazwa formatu MQ komunikatu odpowiedzi	ReplyToFormat
<i>RemoteSysId</i>	Identyfikator zdalnego systemu CICS do użycia	IdentyfikatorRemoteSys

Tabela 22. Pola w tabeli MQCIH (kontynuacja)		
Pole	Opis	Temat
<i>RemoteTransId</i>	CICS RTRANSID do użycia	RemoteTransId
<i>TransactionId</i>	Transakcja do dołączenia	TransactionId
<i>FacilityLike</i>	Emulowane atrybuty terminala	FacilityLike
<i>AttentionId</i>	Klawisz AID	AttentionId
<i>StartCode</i>	Kod początkowy transakcji	StartCode
<i>CancelCode</i>	Abend-kod transakcji	CancelCode
<i>NextTransactionId</i>	Następna transakcja do dołączenia	NextTransactionId
<i>Reserved2</i>	Zarezerwowane	Reserved2
<i>Reserved3</i>	Zarezerwowane	Reserved3
Uwaga: Pozostałe pola nie są obecne, jeśli wartość <i>Version</i> jest mniejsza niż MQCIH_VERSION_2.		
<i>CursorPosition</i>	Pozycja kursora	CursorPosition
<i>ErrorOffset</i>	Przesunięcie błędu w komunikacie	ErrorOffset
<i>InputItem</i>	Zarezerwowane	InputItem
<i>Reserved4</i>	Zarezerwowane	Reserved4

Przegląd produktu MQCIH

Struktura MQCIH opisuje informacje nagłówka dla komunikatu wysłanego do programu CICS w CICS bridge. W przypadku dowolnej platformy IBM MQ obsługiwanej przez użytkownika można utworzyć i przesłać komunikat zawierający strukturę MQCIH, ale tylko menedżer kolejek produktu IBM MQ for z/OS może używać CICS bridge. Oznacza to, że aby komunikat mógł zostać wysłany do programu CICS z menedżera kolejek innego niż z/OS, sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek produktu z/OS, przez który komunikat może być kierowany.

Dostępność: AIX, HP-UX, z/OS, Solaris, Linux, Windows oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQCIH opisuje informacje, które mogą być obecne na początku komunikatu wysyłanego do serwera CICS bridge za pomocą programu IBM MQ for z/OS.

Nazwa formatu: MQFMT_CICS.

Wersja: Bieżąca wersja produktu MQCIH to MQCIH_VERSION_2. Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach, które są następujące.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQCIH, przy czym wartość początkowa pola *Version* jest ustawiona na MQCIH_VERSION_2.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQCIH i danych komunikatu aplikacji:

- Aplikacje, które łączą się z menedżerem kolejek, do którego należy kolejka CICS bridge, muszą udostępniać strukturę MQCIH, która znajduje się w zestawie znaków i kodowaniu menedżera kolejek. Wynika to z faktu, że konwersja danych struktury MQCIH nie jest wykonywana w tym przypadku.
- Aplikacje, które łączą się z innymi menedżerami kolejek, mogą udostępniać strukturę MQCIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań; odbierający agent kanału komunikatów połączony z menedżerem kolejek, do którego należy kolejka CICS bridge, przekształca strukturę MQCIH.

- Dane komunikatu aplikacji zgodnie ze strukturą MQCIH muszą być w tym samym zestawie znaków i kodowaniu co struktura MQCIH. Nie można używać pól *CodedCharSetId* i *Encoding* w strukturze MQCIH, aby określić zestaw znaków i kodowanie danych komunikatu aplikacji.

Należy podać wyjście konwersji danych, aby przekształcić dane komunikatu aplikacji, jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek.

Użycie: Jeśli aplikacja wymaga wartości, które są takie same, jak początkowe wartości wyświetlane w programie Tabela 24 na stronie 313, a most jest uruchomiony z wartością AUTH=LOCAL lub AUTH=IDENTIFY, można pominąć strukturę MQCIH z komunikatu. We wszystkich innych przypadkach struktura musi być obecna.

Most akceptuje strukturę MQCIH w wersji version-1 lub version-2, ale w przypadku transakcji 3270 musi być używana struktura version-2.

Aplikacja musi upewnić się, że pola udokumentowane jako pola żądania mają odpowiednie wartości w komunikacie wystanym do mostu. Pola te są danymi wejściowymi do mostu.

Pola udokumentowane jako pola odpowiedzi są ustawiane przez CICS bridge w komunikacie odpowiedzi, który most wysyła do aplikacji. Informacje o błędach są zwracane w polach *ReturnCode*, *Function*, *CompCode*, *Reason* i *AbendCode*, ale nie wszystkie z nich są ustawiane we wszystkich przypadkach. Tabela 23 na stronie 304 pokazuje, które pola są ustawione dla różnych wartości *ReturnCode*.

Tabela 23. Zawartość pól informacji o błędzie w strukturze MQCIH dla MQCIH				
ReturnCode	Function	CompCode	Reason	AbendCode
MQCRC_OK	-	-	-	-
MQCRC_BRIDGE_ERROR	-	-	MQFB_CICS_*	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	Nazwa wywołania MQ	MQ CompCode	MQ Reason	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	CICS ABCODE

Pola dla MQCIH

Struktura MQCIH zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**.

AbendCode (MQCHAR4)

AbendCode to pole odpowiedzi. Długość tego pola jest określona przez wartość MQ_ABEND_CODE_LENGTH. Początkowa wartość tego pola to 4 puste znaki.

Wartość zwracana w tym polu jest istotna tylko wtedy, gdy w polu *ReturnCode* znajduje się wartość MQCRC_APPLICATION_ABEND lub MQCRC_BRIDGE_ABEND. Jeśli tak się stanie, *AbendCode* zawiera wartość CICS ABCODE.

Deskryptor ADSDescriptor (MQLONG)

To pole jest wskaźnikiem określaniem, czy deskryptory ADS mają być wysyłane na żądania SEND i RECEIVE BMS.

Zdefiniowane są następujące wartości:

MQCADSD_NONE

Nie wysyłaj ani nie odbieraj deskryptorów ADS.

MQCADSD_SEND,

Wyślij deskryptory ADS.

MQCADSD_RECV

Odbieranie deskryptorów ADS.

MQCADSD_MSGFORMAT

Użyj formatu komunikatu dla deskryptorów ADS.

Powoduje to wysyłanie lub odbieranie deskryptorów ADS przy użyciu długiej postaci deskryptora ADS. Długi formularz zawiera pola wyrównane w granicach 4-bajtowych.

Ustaw pole *ADSDescriptor* w następujący sposób:

- Jeśli nie korzystasz z deskryptorów ADS, ustaw pole na wartość MQCADSD_NONE.
- Jeśli w każdym środowisku używane są deskryptory ADS z *tym samym* identyfikatorem CCSID, należy ustawić to pole na sumę wartości MQCADSD_SEND i MQCADSD_RECV.
- Jeśli w każdym środowisku używane są deskryptory ADS z *różnymi* identyfikatorami CCSID, należy ustawić pole na sumę wartości MQCADSD_SEND, MQCADSD_RECV i MQCADSD_MSGFORMAT.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest MQCADSD_NONE.

AttentionId (MQCHAR4)

Wartość w tym polu określa początkową wartość klucza AID podczas uruchamiania transakcji. Jest to wartość 1 bajtowa, wyrównana do lewej strony.

AttentionId -pole żądania używane tylko w przypadku transakcji 3270. Długość tego pola jest podana przez wartość MQ_ATTENTION_ID_LENGTH. Początkowa wartość tego pola to cztery znaki odstępu.

Element uwierzytelniający (MQCHAR8)

Wartością tego pola jest hasło lub passticket.

Jeśli uwierzytelnianie za pomocą identyfikatora użytkownika jest aktywne dla partycji CICS bridge, produkt *Authenticator* jest używany z identyfikatorem użytkownika w kontekście tożsamości MQMD w celu uwierzytelnienia nadawcy komunikatu.

To jest pole żądania. Długość tego pola jest podana przez parametr MQ_AUTHENTICATOR_LENGTH. Początkowa wartość tego pola wynosi 8 znaków odstępu.

CancelCode (MQCHAR4)

Wartość w tym polu jest kodem abend używanym do zakończenia transakcji (zwykle jest to transakcja konwersacyjna, która żąda większej ilości danych). W przeciwnym razie to pole jest puste.

To pole jest polem żądania używanym tylko dla transakcji 3270. Długość tego pola jest podana przez wartość MQ_CANCEL_CODE_LENGTH. Początkowa wartość tego pola to cztery znaki odstępu.

CodedCharSetId (MQLONG)

CodedCharSetId jest polem zastrzeżonym; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

Identyfikator zestawu znaków dla obsługiwanych struktur, które są zgodne ze strukturą MQCIH, jest taki sam, jak identyfikator zestawu znaków w strukturze MQCIH i jest brany z dowolnego poprzedzającego nagłówka IBM MQ .

CompCode (MQLONG)

To pole jest polem odpowiedzi. Jego początkowa wartość to MQCC_OK

Wartość zwracana w tym polu zależy od *ReturnCode* . patrz [Tabela 23 na stronie 304](#).

ConversationalTask (MQLONG)

To pole służy do określania, czy należy zezwolić na wydawanie żądań dla zadania, czy też zatrzymać zadanie i wydać komunikat o błędzie.

Wartość musi być jedną z następujących opcji:

MQCCT_YES

Zadanie jest konwersacyjne.

MQCCT_NO

Zadanie nie jest konwersacyjne.

To pole jest polem żądania używanym tylko dla transakcji 3270. Początkowa wartość tego pola to MQCCT_NO.

CursorPosition (MQLONG)

Wartość w tym polu powoduje wyświetlenie początkowej pozycji kursora po uruchomieniu transakcji. W przypadku transakcji konwersacyjnych pozycja kursora znajduje się w wektorze RECEIVE.

To pole jest polem żądania używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

Kodowanie (MQLONG)

To pole jest polem zastrzeżonym; jego wartość nie jest znacząca. Jego wartością początkową jest 0.

Kodowanie obsługiwanych struktur, które są zgodne ze strukturą MQCIH, jest takie samo, jak kodowanie samej struktury MQCIH i pobierane z dowolnego poprzedzającego nagłówka IBM MQ .

ErrorOffset (MQLONG)

W polu ErrorOffset jest wyświetlana pozycja niepoprawnych danych wykrytych przez wyjście mostu. To pole udostępnia przesunięcie od początku komunikatu do miejsca położenia niepoprawnych danych.

ErrorOffset jest polem odpowiedzi używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

Narzędzie (MQBYTE8)

W tym polu jest wyświetlany 8-bajtowy znacznik obiektu mostu.

Znacznik obiektu pomostowego umożliwia wykonywanie wielu transakcji w pseudo konwersacji w celu użycia tego samego obiektu pomostowego (wirtualnego terminalu 3270). W przypadku pierwszego lub tylko komunikatu w pseudo konwersacji ustaw wartość MQCFAC_NONE. Ta wartość informuje program CICS o przydzielaniu nowego obiektu mostu dla tego komunikatu. Znacznik obiektu pomostowego jest zwracany w komunikatach odpowiedzi, gdy w komunikacie wejściowym podano niezerową wartość *FacilityKeepTime* . Kolejne komunikaty wejściowe w pseudo-konwersacji muszą następnie używać tego samego znacznika obiektu pomostowego.

Zdefiniowane są następujące wartości specjalne:

MQCFAC_NONE

Nie określono znacznika obiektu.

W przypadku języka programowania w języku C stała wartość MQCFAC_NONE_ARRAY jest również zdefiniowana i ma taką samą wartość jak MQCFAC_NONE, ale jest tablicą znaków zamiast łańcucha.

To pole jest zarówno polem żądania, jak i polem odpowiedzi używanym tylko dla transakcji 3270. Długość tego pola jest podana przez parametr MQ_FACILITY_LENGTH. Wartością początkową tego pola jest MQCFAC_NONE.

Czas przechowywania FacilityKeep(MQLONG)

FacilityKeep: Czas (w sekundach), przez który obiekt mostu jest przechowywany po zakończeniu transakcji użytkownika.

W przypadku transakcji pseudo-konwersacyjnych należy określić wartość, która odpowiada oczekiwanowi czasu trwania pseudo-konwersacji; wartość zero dla ostatniej transakcji pseudo-konwersacji, a dla innych typów transakcji-wartość zero.

To pole jest polem żądania używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0.

FacilityLike (MQCHAR4)

FacilityLike to nazwa zainstalowanego terminalu, który ma być używany jako model dla obiektu pomostowego.

Wartość pusta oznacza, że *FacilityLike* jest pobierana z definicji profilu transakcji mostu lub używana jest wartość domyślna.

To pole jest polem żądania używanym tylko dla transakcji 3270. Długość tego pola jest podana przez parametr MQ_FACILITY_LIKE_LENGTH. Początkowa wartość tego pola to cztery znaki odstępu.

Flagi (MQLONG)

To pole jest polem żądania. Wartością początkową tego pola jest MQCIH_NONE.

Wartość musi być następująca:

MQCIH_NONE

Brak flag.

MQCIH_PASS_EXPIRATION

Komunikat odpowiedzi zawiera:

- Te same opcje raportu utraty ważności, jak w przypadku komunikatu żądania.
- Pozostały czas utraty ważności z komunikatu żądania bez korekty z powodu czasu przetwarzania mostu.

Jeśli ta wartość zostanie pominięta, czas utraty ważności jest ustawiany na *nieograniczony*.

MQCIH_REPLY_WITHOUT_NULLS,

Długość komunikatu odpowiedzi dla żądania programu CICS DPL jest dopasowywana w taki sposób, aby wykluczać końcowe znaki puste (X'00 ') na końcu obszaru COMMAREA zwróconego przez program DPL. Jeśli ta wartość nie zostanie ustawiona, wartości NULL mogą być znaczące, a zwracana jest pełna komenda COMMAREA.

MQCIH_SYNC_ON_RETURN

Odsyłacz CICS dla żądań DPL korzysta z opcji SYNCONRETURN, co powoduje, że program CICS ma przyjąć punkt synchronizacji po zakończeniu działania programu, jeśli jest on dostarczany do innego regionu produktu CICS. Most nie określa, do którego regionu CICS ma zostać wysłane żądanie, które jest kontrolowane przez definicję programu CICS lub urządzenia do równoważenia obciążenia.

Format (MQCHAR8)

W tym polu wyświetlana jest nazwa formatu IBM MQ danych, które są zgodne ze strukturą MQCIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak reguły kodowania pola *Format* w strukturze MQMD.

Ta nazwa formatu jest również używana dla komunikatu odpowiedzi, jeśli pole *ReplyToFormat* ma wartość MQFMT_NONE.

- W przypadku żądań DPL *Format* musi być nazwą formatu komendy COMMAREA.
- W przypadku żądań 3270 *Format* musi mieć wartość CSQCBDCI, a most ustawia format na CSQCBDCO dla komunikatów odpowiedzi.

Wyjścia konwersji danych dla tych formatów muszą być zainstalowane w menedżerze kolejek, w którym mają być uruchomione.

Jeśli komunikat żądania wygeneruje komunikat o błędzie, komunikat odpowiedzi o błędzie ma format nazwy MQFMT_STRING.

To pole jest polem żądania. Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Funkcja (MQCHAR4)

To pole jest polem odpowiedzi. Długość tego pola jest podana przez parametr MQ_FUNCTION_LENGTH. Wartością początkową tego pola jest MQCFUNC_NONE.

Wartość zwracana w tym polu zależy od *ReturnCode* . patrz Tabela 23 na stronie 304. Następujące wartości są możliwe, gdy *Function* zawiera nazwę wywołania IBM MQ :

MQCFUNC_MQCONN

Wywołanie MQCONN.

MQCFUNC_MQGET-MQGET

Wywołanie MQGET.

MQCFUNC_MQINQ

Wywołanie MQINQ.

MQCFUNC_MQOPEN

Wywołanie MQOPEN.

MQCFUNC_MQPUT MQPUT

Wywołanie MQPUT.

MQCFUNC_MQPUT1

Wywołanie MQPUT1 .

MQCFUNC_NONE

Brak połączenia.

We wszystkich przypadkach dla języka programowania C zdefiniowane są również stałe MQCFUNC_*_ARRAY; te stałe mają te same wartości, co odpowiadające im stałe MQCFUNC_* , ale są tablicami znaków zamiast łańcuchów.

Przedział czasu GetWait(MQLONG)

To pole jest polem żądania. Jego początkowa wartość to MQCGWI_DEFAULT.

To pole ma zastosowanie tylko wtedy, gdy parametr *UOWControl* ma wartość MQCUOWC_FIRST. Umożliwia on aplikacji wysyłającej określenie przybliżonego czasu (w milisekundach), przez który wywołania MQGET wydane przez most będą oczekiwać na drugie i kolejne komunikaty żądań dla jednostki pracy uruchomionej przez ten komunikat. Ta funkcja przesłania domyślny przedział czasu oczekiwania używany przez most. Można użyć następujących wartości specjalnych:

MQCGWI_DEFAULT

Domyślny interwał oczekiwania.

Ta wartość powoduje, że program CICS bridge będzie oczekiwał na czas określony podczas uruchamiania mostu.

MQWI_UNLIMITED

Nieograniczony przedział czasu oczekiwania.

InputItem (MQLONG)

To pole jest polem zastrzeżonym. Wartość musi być równa 0.

To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

LinkType (MQLONG)

To pole jest polem żądania. Jego początkowa wartość to MQCLT_PROGRAM.

Ta wartość wskazuje typ obiektu, z którym most próbuje się połączyć. Musi to być jedna z następujących wartości:

PROGRAM MQCLT_PROGRAM

Program DPL.

MQCLT_TRANSACTION,

Transakcja 3270.

Identyfikator NextTransaction(MQCHAR4)

Ta wartość jest nazwą kolejnej transakcji zwracanej przez transakcję użytkownika (zwykle przez EXEC CICS RETURN TRANSID). Jeśli nie ma następnej transakcji, to pole jest puste.

To pole jest polem odpowiedzi używanym tylko dla transakcji 3270. Długość tego pola jest podana przez wartość `MQ_TRANSACTION_ID_LENGTH`. Początkowa wartość tego pola to cztery znaki odstępu.

Długość OutputData(MQLONG)

To pole jest polem żądania używanym tylko dla programów DPL. Jego początkowa wartość to `MQCODL_AS_INPUT`.

Ta wartość określa długość danych użytkownika, które mają zostać zwrócone do klienta w komunikacie odpowiedzi. Ta długość obejmuje 8-bajtową nazwę programu. Długość pola `COMMAREA` przekazana do programu dowiązanego jest wartością maksymalną tego pola i długością danych użytkownika w komunikacie żądania, pomniejszonym o 8.

Uwaga: Długość danych użytkownika w komunikacie jest długością komunikatu wykluczając strukturę `MQCIH`.

Jeśli długość danych użytkownika w komunikacie żądania jest mniejsza niż *OutputDataLength*, używana jest opcja `DATALENGTH` komendy `LINK`, która umożliwia sprawne działanie funkcji `LINK` w innym regionie CICS.

Można użyć następującej wartości specjalnej:

MQCODL_AS_INPUT

Długość danych wyjściowych jest taka sama jak długość danych wejściowych.

Ta wartość może być potrzebna, nawet jeśli nie jest wymagana żadna odpowiedź, w celu zapewnienia, że komenda przekazana do programu połączonego jest wystarczająca.

Przyczyna (MQLONG)

To pole jest polem odpowiedzi. Jego początkowa wartość to `MQRC_NONE`.

Wartość zwracana w tym polu zależy od *ReturnCode*. patrz [Tabela 23 na stronie 304](#).

Identyfikator RemoteSys(MQCHAR4)

W tym polu wyświetlany jest identyfikator systemu CICS systemu CICS przetwarzającego żądanie.

Jeśli to pole jest puste, żądanie systemowe CICS jest przetwarzane w tym samym systemie CICS, co monitor mostu. Użyty identyfikator `SYSID` jest zwracany w komunikacie odpowiedzi.

W przypadku pseudo-konwersacji 3270 wszystkie kolejne komunikaty w konwersacji muszą określać zdalny identyfikator `SYSID` zwrócony w odpowiedzi początkowej. Jeśli zostanie podany, identyfikator `SYSID` musi być następujący:

- Bądź aktywny.
- Mają dostęp do kolejki żądań IBM MQ.
- Są one dostępne dla łączy ISC CICS z systemu CICS monitora mostu.

Identyfikator RemoteTrans(MQCHAR4)

To pole jest opcjonalnym polem żądania. Długość tego pola jest podana przez wartość `MQ_TRANSACTION_ID_LENGTH`.

Jeśli to pole zostanie określone, pole będzie używane jako wartość parametru `RTRANSID` produktu CICS `START`.

Format ReplyTo(MQCHAR8)

Wartość w tym polu jest nazwą formatu IBM MQ komunikatu odpowiedzi, który jest wysyłany w odpowiedzi na bieżący komunikat.

Reguły kodowania tego pola są takie same, jak reguły kodowania pola *Format* w strukturze `MQMD`.

To pole jest polem żądania używanym tylko dla programów DPL. Długość tego pola jest podana przez wartość `MQ_FORMAT_LENGTH`. Wartością początkową tego pola jest `MQFMT_NONE`.

Reserved1 (MQCHAR8)

To pole jest polem zastrzeżonym. Wartość musi zawierać 8 odstępów.

Reserved2 (MQCHAR8)

To pole jest polem zastrzeżonym. Wartość musi zawierać 8 odstępów.

Reserved3 (MQCHAR8)

To pole jest polem zastrzeżonym. Wartość musi zawierać 8 odstępów.

Reserved4 (MQLONG)

To pole jest polem zastrzeżonym. Wartość musi być równa 0.

To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

ReturnCode (MQLONG)

Wartość w tym polu jest kodem powrotu z CICS bridge opisującym wynik przetwarzania wykonywanego przez most. To pole jest polem odpowiedzi z początkową wartością MQCRC_OK.

Pola *Function*, *CompCode*, *Reason* i *AbendCode* mogą zawierać dodatkowe informacje (patrz [Tabela 23 na stronie 304](#)). Wartość ta jest jedną z następujących wartości:

MQCRC_APPLICATION_ABEND

(5, X'005 ') Aplikacja zakończyła się nieprawidłowo.

MQCRC_BRIDGE_ABEND

(4, X'004 ') CICS bridge zakończyła się nieprawidłowo.

MQCRC_BRIDGE_ERROR

(3, X'003 ') CICS bridge wykrył błąd.

MQCRC_BRIDGE_TIMEOUT

(8, X'008 ') Komunikat drugi lub późniejszy w bieżącej jednostce pracy nie został odebrany w określonym czasie.

MQCRC_CICS_EXEC_ERROR

(1, X'001 ') Instrukcja EXEC CICS wykryła błąd.

BŁĄD MQCRC_MQ_API_ERROR

(2, X'002 ') Wywołanie programu MQ wykryło błąd.

MQCRC_OK

(0, X'000 ') Brak błędu.

MQCRC_PROGRAM_NOT_AVAILABLE

(7, X'007 ') Program nie jest dostępny.

MQCRC_SECURITY_ERROR,

(6, X'006 ') Wystąpił błąd zabezpieczeń.

MQCRC_TRANSID_NOT_AVAILABLE

(9, X'009 ') Transakcja nie jest dostępna.

StartCode (MQCHAR4)

Wartość tego pola to indyktor określający, czy most emuluje transakcję terminalową, czy transakcję zainicjowaną z parametrem START.

Wartość musi być jedną z następujących wartości:

MQCSC_START

Uruchom.

MQCSC_STARTDATA,

Uruchom dane.

MQCSC_TERMINPUT

Wejście terminalu.

MQCSC_NONE

Brak.

We wszystkich przypadkach dla języka programowania C definiowane są także stałe MQCSC_*_ARRAY; te stałe mają te same wartości, co odpowiadające im stałe MQCSC_*, ale są tablicami znaków zamiast łańcuchów.

W odpowiedzi z mostu pole to jest ustawione na kod początkowy odpowiedni dla następnego identyfikatora transakcji zawartego w polu *NextTransactionId*. W odpowiedzi możliwe są następujące kody początkowe:

- MQCSC_START
- MQCSC_STARTDATA,
- MQCSC_TERMINPUT

W przypadku serwera transakcji CICS w wersji 1.2 to pole jest tylko polem żądania; jego wartość w odpowiedzi jest niezdefiniowana.

W przypadku produktu CICS Transaction Server w wersji 1.3 i kolejnych wersji to pole jest zarówno polem żądania, jak i pola odpowiedzi.

To pole jest używane tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_START_CODE_LENGTH. Wartością początkową tego pola jest MQCSC_NONE.

StrucId (MQCHAR4)

To pole jest polem żądania, którego początkowa wartość to MQCIH_STRUC_ID.

Wartość musi być następująca:

MQCIH_STRUC_ID

Identyfikator struktury nagłówka informacji CICS.

Dla języka programowania C zdefiniowana jest również stała MQCIH_STRUC_ID_ARRAY; ma taką samą wartość jak MQCIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucLength (MQLONG)

To pole jest polem żądania, którego początkowa wartość to MQCIH_LENGTH_2.

Wartość musi być jedną z następujących wartości:

MQCIH_LENGTH_1

Długość struktury nagłówka informacyjnego produktu version-1 CICS.

MQCIH_LENGTH_2

Długość struktury nagłówka informacji o nazwie version-2 CICS.

Następująca stała określa długość bieżącej wersji:

MQCIH_CURRENT_LENGTH

Długość bieżącej wersji struktury nagłówka informacyjnego produktu CICS.

Status TaskEnd (MQLONG)

To pole jest polem odpowiedzi, w którym wyświetlany jest status transakcji użytkownika na końcu zadania. To pole jest używane tylko dla transakcji 3270, a jego wartością początkową jest MQCTES_NOSYNC.

Zwracana jest jedna z następujących wartości:

MQCTES_NOSYNC

Niezsynchronizowane.

Transakcja użytkownika nie została jeszcze zakończona i nie została zsynchronizowana. W tym przypadku pole *MsgType* w strukturze MQMD to MQMT_REQUEST.

MQCTES_COMMIT

Zatwierdź jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona, ale została ona zsynchronizowana z pierwszą jednostką pracy. Pole *MsgType* w strukturze MQMD to MQMT_DATAGRAM w tym przypadku.

MQCTES_BACKOUT

Wycofuje jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona. Wycofana jest bieżąca jednostka pracy. Pole *MsgType* w strukturze MQMD to MQMT_DATAGRAM w tym przypadku.

MQCTES_ENDTASK

Zadanie zakończenia.

Transakcja użytkownika została zakończona (lub została zakończona abkończyniem). Pole *MsgType* w strukturze MQMD to MQMT_REPLY w tym przypadku.

TransactionId (MQCHAR4)

To pole jest polem żądania. Jej długość jest nadawana przez wartość MQ_TRANSACTION_ID_LENGTH. Początkowa wartość tego pola to cztery znaki odstępu.

Jeśli parametr *LinkType* ma wartość MQCLT_TRANSACTION, *TransactionId* jest identyfikatorem transakcji użytkownika, który ma zostać uruchomiony. W tym przypadku należy podać niepustą wartość.

Jeśli parametr *LinkType* ma wartość MQCLT_PROGRAM, *TransactionId* jest kodem transakcji, w ramach którego mają być uruchomione wszystkie programy w jednostce pracy. Jeśli zostanie podana pusta wartość, zostanie użyty domyślny kod transakcji mostu DPL CICS (CKBP). Jeśli wartość jest niepusta, należy ją zdefiniować jako CICS jako transakcję lokalną przy użyciu programu początkowego, który jest CSQCBP00. To pole ma zastosowanie tylko wtedy, gdy parametr *UOWControl* ma wartość MQCUOWC_FIRST lub MQCUOWC_ONLY.

UOWControl (MQLONG)

To pole jest polem żądania, które steruje przetwarzaniem jednostki pracy wykonywaną przez CICS bridge. Wartością początkową tego pola jest MQCUOWC_ONLY.

Można zażądać utworzenia mostu w celu uruchomienia pojedynczej transakcji lub jednego lub większej liczby programów w ramach jednostki pracy. Pole wskazuje, czy produkt CICS bridge uruchamia jednostkę pracy, wykonuje żadaną funkcję w bieżącej jednostce pracy, czy kończy jednostkę pracy, zatwierdzając ją lub wycofując z niej. Obsługiwane są różne kombinacje, aby zoptymalizować przepływy transmisji danych.

Wartość musi być jedną z następujących wartości:

MQCUOWC_ONLY

Uruchom jednostkę pracy, wykonaj funkcję, a następnie zatwierdź jednostkę pracy.

MQCUOWC_CONTINUE

Dodatkowe dane dla bieżącej jednostki pracy (tylko 3270).

MQCUOWC_FIRST

Uruchom jednostkę pracy i wykonaj funkcję.

MQCUOWC_MIDDLE

Wykonywanie funkcji w bieżącej jednostce pracy

MQCUOWC_LAST

Wykonaj funkcję, a następnie zatwierdź jednostkę pracy.

MQCUOWC_COMMIT

Zatwierdź jednostkę pracy (tylko DPL).

MQCUOWC_BACKOUT

Wycofuje się z jednostki pracy (tylko DPL).

Wersja (MQLONG)

To pole jest polem żądania. Jego początkowa wartość to MQCIH_VERSION_2.

Wartość musi być jedną z następujących wartości:

MQCIH_VERSION_1

Struktura nagłówka informacji Version-1 CICS .

MQCIH_VERSION_2

Struktura nagłówka informacji Version-2 CICS .

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MQCIH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka informacyjnego produktu CICS .

Wartości początkowe i deklaracje języków dla MQCIH

Tabela 24. Początkowe wartości pól w MQCIH dla MQCIH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQCIH_STRUC_ID	'CIH'
<i>Version</i>	MQCIH_VERSION_2	2
<i>StrucLength</i>	MQCIH_LENGTH_2	180
<i>Encoding</i>	Brak	0
<i>CodedCharSetId</i>	Brak	0
<i>Format</i>	MQFMT_NONE	Puste
<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>	PROGRAM MQCLT_PROGRAM	1
<i>OutputDataLength</i>	MQCODL_AS_INPUT	-1
<i>FacilityKeepTime</i>	Brak	0
<i>ADSDescriptor</i>	MQCADSD_NONE	0
<i>ConversationalTask</i>	MQCCT_NO	0
<i>TaskEndStatus</i>	MQCTES_NOSYNC	0
<i>Facility</i>	MQCFAC_NONE	Wartości null
<i>Function</i>	MQCFUNC_NONE	Puste
<i>AbendCode</i>	Brak	Puste
<i>Authenticator</i>	Brak	Puste
<i>Reserved1</i>	Brak	Puste
<i>ReplyToFormat</i>	MQFMT_NONE	Puste
<i>RemoteSysId</i>	Brak	Puste
<i>RemoteTransId</i>	Brak	Puste
<i>TransactionId</i>	Brak	Puste
<i>FacilityLike</i>	Brak	Puste

Tabela 24. Początkowe wartości pól w MQCIH dla MQCIH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>AttentionId</i>	Brak	Puste
<i>StartCode</i>	MQCSC_NONE	Puste
<i>CancelCode</i>	Brak	Puste
<i>NextTransactionId</i>	Brak	Puste
<i>Reserved2</i>	Brak	Puste
<i>Reserved3</i>	Brak	Puste
<i>CursorPosition</i>	Brak	0
<i>ErrorOffset</i>	Brak	0
<i>InputItem</i>	Brak	0
<i>Reserved4</i>	Brak	0

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraParametr MQCIH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
    MQCHAR4 StrucId;           /* Structure identifier */
    MQLONG Version;           /* Structure version number */
    MQLONG StrucLength;       /* Length of MQCIH structure */
    MQLONG Encoding;         /* Reserved */
    MQLONG CodedCharSetId;    /* Reserved */
    MQCHAR8 Format;           /* MQ format name of data that follows
                             MQCIH */
    MQLONG Flags;             /* Flags */
    MQLONG ReturnCode;        /* Return code from bridge */
    MQLONG CompCode;         /* MQ completion code or CICS EIBRESP */
    MQLONG Reason;           /* MQ reason or feedback code, or CICS
                             EIBRESP2 */
    MQLONG UOWControl;        /* Unit-of-work control */
    MQLONG GetWaitInterval;   /* Wait interval for MQGET call issued
                             by bridge task */
    MQLONG LinkType;         /* Link type */
    MQLONG OutputDataLength;  /* Output COMMAREA data length */
    MQLONG FacilityKeepTime;  /* Bridge facility release time */
    MQLONG ADSDescriptor;     /* Send/receive ADS descriptor */
    MQLONG ConversationalTask; /* Whether task can be conversational */
    MQLONG TaskEndStatus;     /* Status at end of task */
    MQBYTE8 Facility;        /* Bridge facility token */
    MQCHAR4 Function;        /* MQ call name or CICS EIBFN
                             function */
    MQCHAR4 AbendCode;        /* Abend code */
    MQCHAR8 Authenticator;    /* Password or passticket */
    MQCHAR8 Reserved1;       /* Reserved */
    MQCHAR8 ReplyToFormat;    /* MQ format name of reply message */
    MQCHAR4 RemoteSysId;     /* Reserved */
    MQCHAR4 RemoteTransId;   /* Reserved */
    MQCHAR4 TransactionId;    /* Transaction to attach */
    MQCHAR4 FacilityLike;     /* Terminal emulated attributes */
    MQCHAR4 AttentionId;     /* AID key */
    MQCHAR4 StartCode;       /* Transaction start code */
};
```

```

MQCHAR4 CancelCode;          /* Abend transaction code */
MQCHAR4 NextTransactionId;    /* Next transaction to attach */
MQCHAR8 Reserved2;           /* Reserved */
MQCHAR8 Reserved3;           /* Reserved */
MQLONG CursorPosition;        /* Cursor position */
MQLONG ErrorOffset;           /* Offset of error in message */
MQLONG InputItem;             /* Reserved */
MQLONG Reserved4;             /* Reserved */
};

```

Deklaracja języka COBOL

```

** MQCIH structure
10 MQCIH.
** Structure identifier
15 MQCIH-STRUCID PIC X(4).
** Structure version number
15 MQCIH-VERSION PIC S9(9) BINARY.
** Length of MQCIH structure
15 MQCIH-STRUCLength PIC S9(9) BINARY.
** Reserved
15 MQCIH-ENCODING PIC S9(9) BINARY.
** Reserved
15 MQCIH-CODEDCHARSETID PIC S9(9) BINARY.
** MQ format name of data that follows MQCIH
15 MQCIH-FORMAT PIC X(8).
** Flags
15 MQCIH-FLAGS PIC S9(9) BINARY.
** Return code from bridge
15 MQCIH-RETURNCODE PIC S9(9) BINARY.
** MQ completion code or CICS EIBRESP
15 MQCIH-COMPCODE PIC S9(9) BINARY.
** MQ reason or feedback code, or CICS EIBRESP2
15 MQCIH-REASON PIC S9(9) BINARY.
** Unit-of-work control
15 MQCIH-UOWCONTROL PIC S9(9) BINARY.
** Wait interval for MQGET call issued by bridge task
15 MQCIH-GETWAITINTERVAL PIC S9(9) BINARY.
** Link type
15 MQCIH-LINKTYPE PIC S9(9) BINARY.
** Output COMMAREA data length
15 MQCIH-OUTPUTDATALENGTH PIC S9(9) BINARY.
** Bridge facility release time
15 MQCIH-FACILITYKEEPTIME PIC S9(9) BINARY.
** Send/receive ADS descriptor
15 MQCIH-ADSDESCRIPTOR PIC S9(9) BINARY.
** Whether task can be conversational
15 MQCIH-CONVERSATIONALTASK PIC S9(9) BINARY.
** Status at end of task
15 MQCIH-TASKENDSTATUS PIC S9(9) BINARY.
** Bridge facility token
15 MQCIH-FACILITY PIC X(8).
** MQ call name or CICS EIBFN function
15 MQCIH-FUNCTION PIC X(4).
** Abend code
15 MQCIH-ABENDCODE PIC X(4).
** Password or passticket
15 MQCIH-AUTHENTICATOR PIC X(8).
** Reserved
15 MQCIH-RESERVED1 PIC X(8).
** MQ format name of reply message
15 MQCIH-REPLYTOFORMAT PIC X(8).
** Reserved
15 MQCIH-REMOTESYSID PIC X(4).
** Reserved
15 MQCIH-REMOtetransid PIC X(4).
** Transaction to attach
15 MQCIH-TRANSACTIONID PIC X(4).
** Terminal emulated attributes
15 MQCIH-FACILITYLIKE PIC X(4).
** AID key
15 MQCIH-ATTENTIONID PIC X(4).
** Transaction start code
15 MQCIH-STARTCODE PIC X(4).
** Abend transaction code
15 MQCIH-CANCELCODE PIC X(4).
** Next transaction to attach
15 MQCIH-NEXTTRANSACTIONID PIC X(4).
** Reserved

```

```

15 MQCIH-RESERVED2          PIC X(8).
**   Reserved
15 MQCIH-RESERVED3          PIC X(8).
**   Cursor position
15 MQCIH-CURSORPOSITION     PIC S9(9) BINARY.
**   Offset of error in message
15 MQCIH-ERROROFFSET        PIC S9(9) BINARY.
**   Reserved
15 MQCIH-INPUTITEM          PIC S9(9) BINARY.
**   Reserved
15 MQCIH-RESERVED4          PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQCIH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 StrucLength      fixed bin(31),    /* Length of MQCIH structure */
3 Encoding         fixed bin(31),    /* Reserved */
3 CodedCharSetId   fixed bin(31),    /* Reserved */
3 Format            char(8),          /* MQ format name of data that
                                     follows MQCIH */
3 Flags            fixed bin(31),    /* Flags */
3 ReturnCode       fixed bin(31),    /* Return code from bridge */
3 CompCode         fixed bin(31),    /* MQ completion code or CICS
                                     EIBRESP */
3 Reason           fixed bin(31),    /* MQ reason or feedback code, or
                                     CICS EIBRESP2 */
3 UOWControl       fixed bin(31),    /* Unit-of-work control */
3 GetWaitInterval  fixed bin(31),    /* Wait interval for MQGET call
                                     issued by bridge task */
3 LinkType         fixed bin(31),    /* Link type */
3 OutputDataLength fixed bin(31),    /* Output COMMAREA data length */
3 FacilityKeepTime fixed bin(31),    /* Bridge facility release time */
3 ADSDescriptor    fixed bin(31),    /* Send/receive ADS descriptor */
3 ConversationalTask fixed bin(31), /* Whether task can be
                                     conversational */
3 TaskEndStatus    fixed bin(31),    /* Status at end of task */
3 Facility         char(8),          /* Bridge facility token */
3 Function         char(4),          /* MQ call name or CICS EIBFN
                                     function */
3 AbendCode        char(4),          /* Abend code */
3 Authenticator    char(8),          /* Password or passticket */
3 Reserved1        char(8),          /* Reserved */
3 ReplyToFormat    char(8),          /* MQ format name of reply
                                     message */
3 RemoteSysId      char(4),          /* Reserved */
3 RemoteTransId    char(4),          /* Reserved */
3 TransactionId     char(4),          /* Transaction to attach */
3 FacilityLike     char(4),          /* Terminal emulated attributes */
3 AttentionId      char(4),          /* AID key */
3 StartCode        char(4),          /* Transaction start code */
3 CancelCode       char(4),          /* Abend transaction code */
3 NextTransactionId char(4),          /* Next transaction to attach */
3 Reserved2        char(8),          /* Reserved */
3 Reserved3        char(8),          /* Reserved */
3 CursorPosition   fixed bin(31),    /* Cursor position */
3 ErrorOffset      fixed bin(31),    /* Offset of error in message */
3 InputItem        fixed bin(31),    /* Reserved */
3 Reserved4        fixed bin(31);    /* Reserved */

```

Deklaracja High Level Assembler

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

* MQCIH_UOWCONTROL	DS	F	EIBRESP2 Unit-of-work control
MQCIH_GETWAITINTERVAL	DS	F	Wait interval for MQGET call issued by bridge task
* MQCIH_LINKTYPE	DS	F	Link type
MQCIH_OUTPUTDATALENGTH	DS	F	Output COMMAREA data length
MQCIH_FACILITYKEEPTIME	DS	F	Bridge facility release time
MQCIH_ADSDESCRIPTOR	DS	F	Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK	DS	F	Whether task can be conversational
MQCIH_TASKENDSTATUS	DS	F	Status at end of task
MQCIH_FACILITY	DS	XL8	Bridge facility token
MQCIH_FUNCTION	DS	CL4	MQ call name or CICS EIBFN function
MQCIH_ABENDCODE	DS	CL4	Abend code
MQCIH_AUTHENTICATOR	DS	CL8	Password or passticket
MQCIH_RESERVED1	DS	CL8	Reserved
MQCIH_REPLYTOFORMAT	DS	CL8	MQ format name of reply message
MQCIH_REMOTESYSID	DS	CL4	Reserved
MQCIH_REMOTETRANSID	DS	CL4	Reserved
MQCIH_TRANSACTIONID	DS	CL4	Transaction to attach
MQCIH_FACILITYLIKE	DS	CL4	Terminal emulated attributes
MQCIH_ATTENTIONID	DS	CL4	AID key
MQCIH_STARTCODE	DS	CL4	Transaction start code
MQCIH_CANCELCODE	DS	CL4	Abend transaction code
MQCIH_NEXTTRANSACTIONID	DS	CL4	Next transaction to attach
MQCIH_RESERVED2	DS	CL8	Reserved
MQCIH_RESERVED3	DS	CL8	Reserved
MQCIH_CURSORPOSITION	DS	F	Cursor position
MQCIH_ERROROFFSET	DS	F	Offset of error in message
MQCIH_INPUTITEM	DS	F	Reserved
MQCIH_RESERVED4	DS	F	Reserved
* MQCIH_LENGTH	EQU	*-MQCIH	
	ORG	MQCIH	
MQCIH_AREA	DS	CL(MQCIH_LENGTH)	

Wizualna deklaracja podstawowa

Type MQCIH		
StrucId	As String*4	'Structure identifier'
Version	As Long	'Structure version number'
StrucLength	As Long	'Length of MQCIH structure'
Encoding	As Long	'Reserved'
CodedCharSetId	As Long	'Reserved'
Format	As String*8	'MQ format name of data that follows' 'MQCIH'
Flags	As Long	'Flags'
ReturnCode	As Long	'Return code from bridge'
CompCode	As Long	'MQ completion code or CICS EIBRESP'
Reason	As Long	'MQ reason or feedback code, or CICS' 'EIBRESP2'
UOWControl	As Long	'Unit-of-work control'
GetWaitInterval	As Long	'Wait interval for MQGET call issued' 'by bridge task'
LinkType	As Long	'Link type'
OutputDataLength	As Long	'Output COMMAREA data length'
FacilityKeepTime	As Long	'Bridge facility release time'
ADSDescriptor	As Long	'Send/receive ADS descriptor'
ConversationalTask	As Long	'Whether task can be conversational'
TaskEndStatus	As Long	'Status at end of task'
Facility	As MQBYTE8	'Bridge facility token'
Function	As String*4	'MQ call name or CICS EIBFN function'
AbendCode	As String*4	'Abend code'
Authenticator	As String*8	'Password or passticket'
Reserved1	As String*8	'Reserved'
ReplyToFormat	As String*8	'MQ format name of reply message'
RemoteSysId	As String*4	'Reserved'
RemoteTransId	As String*4	'Reserved'
TransactionId	As String*4	'Transaction to attach'
FacilityLike	As String*4	'Terminal emulated attributes'
AttentionId	As String*4	'AID key'
StartCode	As String*4	'Transaction start code'
CancelCode	As String*4	'Abend transaction code'
NextTransactionId	As String*4	'Next transaction to attach'
Reserved2	As String*8	'Reserved'
Reserved3	As String*8	'Reserved'
CursorPosition	As Long	'Cursor position'
ErrorOffset	As Long	'Offset of error in message'
InputItem	As Long	'Reserved'

MQCMHO-opcje tworzenia uchwytu komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 25. Pola w MQCMHO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje

Przegląd produktu MQCMHO

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS and IBM MQ clients.

Cel: Struktura **MQCMHO** umożliwia aplikacjom określanie opcji, które sterują sposobem tworzenia uchwytów komunikatów. Struktura jest parametrem wejściowym w wywołaniu **MQCRTMH**.

Zestaw znaków i kodowanie: Dane w programie **MQCMHO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (**MQENC_NATIVE**).

Pola dla MQCMHO

Struktura MQCMHO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

To pole jest zawsze polem wejściowym. Jego wartością początkową jest MQCMHO_DEFAULT_VALIDATION.

Można określić jedną z następujących opcji:

MQCMHO_VALIDATE

Gdy program **MQSETMP** jest wywoływany w celu ustawienia właściwości w tym uchwycie komunikatu, sprawdzana jest poprawność nazwy właściwości w celu upewnienia się, że:

- nie zawiera niepoprawnych znaków.
- nie rozpoczyna się JMS ani usr.JMS z wyjątkiem następujących:
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType
 - JMSXGroupID
 - JMSXGroupSeq

Te nazwy są zastrzeżone dla właściwości produktu JMS.

- nie jest jednym z następujących słów kluczowych, w żadnej mieszance wielkich lub małych:
 - I
 - Między
 - Escape
 - FAŁSZ
 - zawiera się w
 - JEST

- PODOBNE
- NIE
- NULL
- LUB
- PRAWDA

- nie zaczyna się od ciąa. lub Root. (oprócz elementu Root.MQMD.).

Jeśli właściwość jest zdefiniowana przez produkt MQ(mq. *) i nazwa jest rozpoznawana, pola deskryptora właściwości są ustawiane na poprawne wartości dla właściwości. Jeśli właściwość nie zostanie rozpoznana, pole *Support* w deskrytorze właściwości jest ustawione na wartość **MQPD_OPTIONAL**.

MQCMHO_DEFAULT_VALIDATION

Ta wartość określa, że występuje domyślny poziom sprawdzania poprawności nazw właściwości.

Domyślny poziom sprawdzania poprawności jest równoważny poziomowi określanego przez produkt **MQCMHO_VALIDATE**.

Ta wartość jest wartością domyślną.

MQCMHO_NO_VALIDATION

Nie ma sprawdzania poprawności nazwy właściwości. Patrz opis produktu **MQCMHO_VALIDATE**.

Opcja domyślna: Jeśli nie jest wymagana żadna z powyższych opisanych opcji, można użyć następującej opcji:

MQCMHO_NONE

Wszystkie opcje przyjmują wartości domyślne. Użyj tej wartości, aby wskazać, że nie określono żadnych innych opcji. **MQCMHO_NONE** jest pomocna w dokumentacji programu; nie jest przeznaczona, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

StrucId (MQCHAR4)

To pole jest zawsze polem wejściowym. Jego początkowa wartość to MQCMHO_STRUC_ID.

Jest to identyfikator struktury. Wartość musi być następująca:

MQCMHO_STRUC_ID,

Identyfikator struktury opcji tworzenia uchwytu komunikatu.

Dla języka programowania w języku C jest również zdefiniowana stała **MQCMHO_STRUC_ID_ARRAY**. Ma ona taką samą wartość jak **MQCMHO_STRUC_ID**, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG)

To pole jest zawsze polem wejściowym. Jego początkowa wartość to MQCMHO_VERSION_1.

Jest to numer wersji struktury. Wartość musi być następująca:

MQCMHO_VERSION_1

Version-1 -tworzenie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQCMHO_CURRENT_VERSION

Bieżąca wersja struktury opcji tworzenia uchwytu komunikatu.

Wartości początkowe i deklaracje języków dla MQCMHO

Tabela 26. Początkowe wartości pól w MQCMHO

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQCMHO_STRUC_ID,	'CMHO '
Version	MQCMHO_VERSION_1	1
Options	MQCMHO_DEFAULT_VALIDATION	0

Uwagi:

1. W języku programowania C: zmienna makraParametr MQCMHO_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Deklaracja C

```
struct tagMQCMHO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQCRTMH */
};
```

Deklaracja języka COBOL

```
**      MQCMHO structure
10 MQCMHO.
**      Structure identifier
15 MQCMHO-STRUCID      PIC X(4).
**      Structure version number
15 MQCMHO-VERSION      PIC S9(9) BINARY.
**      Options that control the action of MQCRTMH
15 MQCMHO-OPTIONS      PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQCMHO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options that control the action of MQCRTMH */
```

Deklaracja High Level Assembler

```
MQCMHO          DSECT
MQCMHO_STRUCID  DS   CL4    Structure identifier
MQCMHO_VERSION  DS   F      Structure version number
MQCMHO_OPTIONS  DS   F      Options that control the action of
*                MQCRTMH
MQCMHO_LENGTH   EQU   *-MQCMHO
MQCMHO_AREA     DS   CL(MQCMHO_LENGTH)
```

MQCNO-opcje połączenia

W poniższej tabeli podsumowano pola w strukturze.

Tabela 27. Pola w MQCNO

Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem MQCONN	Opcje
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQCNO_VERSION_2.		
<i>ClientConnOffset</i>	Przesunięcie struktury MQCD dla połączenia klienta	ClientConnPrzesunięcie
<i>ClientConnPtr</i>	Adres struktury MQCD dla połączenia klienckiego	ClientConnPtr
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQCNO_VERSION_3.		
<i>ConnTag</i>	Znacznik połączenia menedżera kolejek	ConnTag
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQCNO_VERSION_4.		
<i>SSLConfigPtr</i>	Adres struktury MQSCO dla połączenia klienckiego	SSLConfigPtr
<i>SSLConfigOffset</i>	Przesunięcie struktury MQSCO dla połączenia klienckiego	SSLConfigOffset
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQCNO_VERSION_5.		
<i>ConnectionId</i>	Unikalny identyfikator połączenia	ConnectionId
<i>SecurityParmsOffset</i>	Parametry bezpieczeństwa	SecurityParmsPrzesunięcie
<i>SecurityParmsPtr</i>	Parametry bezpieczeństwa	SecurityParmsPtr

Informacje pokrewne

Korzystanie z tabeli MQCONN

Przegląd produktu MQCNO

Dostępność: wszystkie wersje z wyjątkiem MQCNO_VERSION_4: AIX, HP-UX, IBM i, Solaris, Linux, Windows, plus IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQCNO umożliwia aplikacji określenie opcji związanych z połączeniem z lokalnym menedżerem kolejek. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQCONN. Więcej informacji na temat korzystania z współużytkowanych uchwytów oraz wywołania MQCONN zawiera sekcja [Połączenia współużytkowane \(niezależne od wątku\) z produktem MQCONN](#).

Wersja: Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję zmaterializowanych tabel zapytań (MQCNO), ale z wartością początkową pola *Version* ustawioną na MQCNO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *Version* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie: Dane w tabeli MQCNO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako IBM MQ MQI client, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQCNO

Struktura MQCNO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ClientConnPrzesunięcie (MQLONG)

ClientConnPrzesunięcie jest przesunięte w bajtach struktury definicji kanału MQCD od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym o wartości początkowej 0.

Opcji *ClientConnOffset* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONN jest uruchomiona jako IBM MQ MQI client. Informacje na temat korzystania z tego pola można znaleźć w opisie pola *ClientConnPtr*.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_2.

ClientConnPtr (MQPTR)

ClientConnPtr jest polem wejściowym. Jego wartością początkową jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Opcji *ClientConnOffset* i *ClientConnPtr* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONN jest uruchomiona jako IBM MQ MQI client. Określając jeden lub drugi z tych pól, aplikacja może sterować definicją kanału połączenia klienckiego, udostępniając strukturę definicji kanału MQCD, która zawiera wymagane wartości.

Jeśli aplikacja jest uruchomiona jako IBM MQ MQI client, ale nie udostępnia struktury MQCD, do wybrania definicji kanału zostanie użyta zmienna środowiskowa MQSERVER. Jeśli parametr MQSERVER nie jest ustawiony, używana jest tabela kanałów klienta.

Jeśli aplikacja nie jest uruchomiona, ponieważ IBM MQ MQI client, *ClientConnOffset* i *ClientConnPtr* są ignorowane.

Jeśli aplikacja udostępnia strukturę MQCD, należy ustawić pola wymienione na wymagane wartości. Pozostałe pola w tabeli MQCD są ignorowane. Łańcuchy znaków można dopełniać spacjami do długości pola lub zakończyć je znakiem o kodzie zero. Więcej informacji na temat pól w strukturze MQCD można znaleźć w sekcji [“Pola” na stronie 1571](#).

Pole w MQCD	Wartość
<i>ChannelName</i>	Nazwa kanału.
<i>Version</i>	Numer wersji struktury. Wartość nie może być mniejsza niż MQCD_VERSION_7.
<i>TransportType</i>	Dowolny obsługiwany typ transportu.
<i>ModeName</i>	Nazwa trybu LU 6.2.
<i>TpName</i>	Nazwa programu transakcyjnego LU 6.2.
<i>SecurityExit</i>	Nazwa wyjścia zabezpieczeń kanału.
<i>SendExit</i>	Nazwa wyjścia wysyłania kanału.
<i>ReceiveExit</i>	Nazwa wyjścia odbierania kanału.
<i>MaxMsgLength</i>	Maksymalna długość w bajtach komunikatów, które mogą być wysyłane za pośrednictwem kanału połączenia klienta.
<i>SecurityUserData</i>	Dane użytkownika dla wyjścia zabezpieczeń.
<i>SendUserData</i>	Dane użytkownika dla wyjścia wysyłania.
<i>ReceiveUserData</i>	Dane użytkownika dla wyjścia odbierania.
<i>UserIdentifier</i>	Identyfikator użytkownika, który ma być używany do ustanawiania sesji LU 6.2.
<i>Password</i>	Hasło, które ma być używane do ustanawiania sesji LU 6.2.
<i>ConnectionName</i>	Nazwa połączenia.

Pole w MQCD	Wartość
<i>HeartbeatInterval</i>	Czas (w sekundach) między przepływami pulsu.
<i>StrucLength</i>	Długość struktury MQCD.
<i>ExitNameLength</i>	Długość nazw wyjść adresowanych przez <i>SendExitPtr</i> i <i>ReceiveExitPtr</i> . Wartość musi być większa od zera, jeśli parametr <i>SendExitPtr</i> lub <i>ReceiveExitPtr</i> jest ustawiony na wartość, która nie jest wskaźnikiem wartości NULL.
<i>ExitDataLength</i>	Długość danych wyjściowych adresowanych przez produkty <i>SendUserDataPtr</i> i <i>ReceiveUserDataPtr</i> . Wartość musi być większa od zera, jeśli parametr <i>SendUserDataPtr</i> lub <i>ReceiveUserDataPtr</i> jest ustawiony na wartość, która nie jest wskaźnikiem wartości NULL.
<i>SendExitsDefined</i>	Liczba wyjść wysyłania skierowanych przez produkt <i>SendExitPtr</i> . Jeśli zero, <i>SendExit</i> i <i>SendUserData</i> , podaj nazwę wyjścia i dane. Jeśli wartość większa niż zero, <i>SendExitPtr</i> i <i>SendUserDataPtr</i> , należy podać nazwy i dane wyjścia, a <i>SendExit</i> i <i>SendUserData</i> muszą być puste.
<i>ReceiveExitsDefined</i>	Liczba wyjść odbierania skierowanych przez produkt <i>ReceiveExitPtr</i> . Jeśli zero, <i>ReceiveExit</i> i <i>ReceiveUserData</i> , podaj nazwę wyjścia i dane. Jeśli wartość większa niż zero, <i>ReceiveExitPtr</i> i <i>ReceiveUserDataPtr</i> , należy podać nazwy i dane wyjścia, a <i>ReceiveExit</i> i <i>ReceiveUserData</i> muszą być puste.
<i>SendExitPtr</i>	Adres imienia pierwszego wyjścia wysyłania.
<i>SendUserDataPtr</i>	Adres danych dla pierwszego wyjścia wysyłania.
<i>ReceiveExitPtr</i>	Adres nazwy pierwszego wyjścia odbierania.
<i>ReceiveUserDataPtr</i>	Adres danych dla pierwszego wyjścia odbierania.
<i>LongRemoteUserIdLength</i>	Długość długiego zdalnego identyfikatora użytkownika.
<i>LongRemoteUserIdPtr</i>	Adres długiego zdalnego identyfikatora użytkownika.
<i>RemoteSecurityId</i>	Identyfikator zabezpieczeń zdalnych.
<i>SSLCipherSpec</i>	SSL CipherSpec.
<i>SSLPeerNamePtr</i>	Adres węzła sieci SSL.
<i>SSLPeerNameLength</i>	Długość nazwy węzła sieci SSL.
<i>KeepAliveInterval</i>	Wartość przekazana do stosu komunikacyjnego dla czasu utrzymywania połączenia dla kanału
<i>LocalAddress</i>	Lokalny adres komunikacyjny, w tym adres IP lokalnego adaptera sieciowego, który ma być używany, oraz zakres portów, które mają być używane dla połączeń wychodzących.

Podaj strukturę definicji kanału na jeden z dwóch sposobów:

- Za pomocą pola przesunięcia *ClientConnOffset*

W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą MQCNO, a następnie strukturę definicji kanału MQCD, a następnie ustawić parametr *ClientConnOffset* na przesunięcie struktury definicji kanału od początku wywołania MQCNO. Upewnij się, że przesunięcie jest poprawne. Parametr *ClientConnPtr* musi być ustawiony na pusty wskaźnik lub zerową liczbę bajtów.

W przypadku języków programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który nie jest przenośny dla różnych środowisk (na przykład w języku programowania COBOL), należy użyć produktu *ClientConnOffset*.

W języku programowania Visual Basic: złożona struktura Tabela MQCNOCD jest dostępna w pliku nagłówkowy CMQXB.BAS; struktura ta zawiera strukturę MQCNO, po której następuje struktura MQCD. Zainicjuj komendę MQCNOCD, wywołując podprocedurę MQCNOCD_DEFAULTS. Produkt MQCNOCD jest używany z MQCONNAny wariant wywołania MQCONN; szczegółowe informacje można znaleźć w opisie wywołania MQCONN.

- Za pomocą pola wskaźnika *ClientConnPtr*

W takim przypadku aplikacja może zadeklarować strukturę definicji kanału oddzielnie od struktury MQCNO, a następnie ustawić wartość *ClientConnPtr* na adres struktury definicji kanału. Ustaw wartość *ClientConnOffset* na zero.

W przypadku języków programowania, które obsługują typ danych wskaźnika w sposób przenośny do różnych środowisk (na przykład język programowania w języku C), należy użyć programu *ClientConnPtr*.

W języku programowania C można użyć zmiennej makra MQCD_CLIENT_CONN_DEFAULT, aby podać początkowe wartości dla struktury, które są bardziej odpowiednie do użycia w wywołaniu MQCONN niż początkowe wartości podane w tabeli MQCD_DEFAULT.

Niezależnie od wybranej techniki, można użyć tylko jednego z następujących produktów: *ClientConnOffset* i *ClientConnPtr*; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CLIENT_CONN_ERROR, jeśli oba są niezerowe.

Po zakończeniu wywołania MQCONN struktura MQCD nie jest przywoływana ponownie.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_2.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtów o wartości all-null.

ConnectionId (MQBYTE24)

ConnectionId to unikalny 24-bajtowy identyfikator, który umożliwia produktowi IBM MQ wiarygodną identyfikację aplikacji. Aplikacja może używać tego identyfikatora do korelacji w wywołaniach PUT i GET. Ten parametr wyjściowy ma wartość początkową równą 24 bajtom o wartości NULL we wszystkich językach programowania.

Menedżer kolejek przypisuje unikalny identyfikator do wszystkich połączeń, niezależnie od tego, czy są one ustanowione. Jeśli obiekt MQCONNX nawiązuje połączenie z MQCNO w wersji 5, aplikacja może określić wartość parametru ConnectionId z zwróconej tabeli MQCNO. Przypisany identyfikator jest gwarantowany jako unikalny wśród wszystkich innych identyfikatorów generowanych przez produkt IBM MQ, takich jak CorrelId, MsgIdi GroupId.

Użyj parametru ConnectionId, aby zidentyfikować długo działające jednostki pracy za pomocą komendy PCF Inquire Connection lub komendy MQSC DISPLAY CONN. Element ConnectionId używany przez komendy MQSC (CONN) jest pochodną wartości ConnectionId zwróconej w tym miejscu. Komendy PCF Inquire i Stop Connection mogą używać identyfikatora ConnectionId zwracanego w tym miejscu bez modyfikacji.

Parametru ConnectionId można użyć w celu wymuszenia zakończenia długotrwałego działania jednostki pracy, określając parametr ConnectionId za pomocą komendy PCF-Zatrzymaj połączenie lub komendy MQSC STOP CONN. Więcej informacji na temat korzystania z tych komend zawiera sekcja [Zatrzymanie połączenia i ZATRZYMANIE](#).

To pole nie jest zwracane, jeśli wersja jest mniejsza niż MQCNO_VERSION_5.

Długość tego pola jest podana przez wartość MQ_CONNECTION_ID_LENGTH.

ConnTag (MQBYTE128)

ConnTag to znacznik, który menedżer kolejek wiąże z zasobami, które mają wpływ na aplikację podczas tego połączenia. Każda instancja aplikacji lub aplikacji musi użyć innej wartości dla znacznika, aby menedżer kolejek mógł poprawnie przekształcać do postaci szeregowej dostęp do odpowiednich zasobów. To pole jest polem wejściowym, a jego wartością początkową jest MQCT_NONE.

Szczegółowe informacje na temat wartości, które mają być używane przez różne aplikacje, zawiera opis opcji MQCNO_*_CONN_TAG_*. Znacznik przestaje być poprawny, gdy aplikacja kończy działanie lub wywołuje wywołanie MQDISC.

Uwaga: Wartości znaczników połączeń rozpoczynające się od MQ w wielkich, dolnych lub mieszanych przypadkach w kodzie ASCII lub EBCDIC są zarezerwowane do użycia przez produkty IBM . Nie należy używać wartości znaczników połączenia rozpoczynających się od tych liter.

Jeśli nie jest wymagany znacznik, należy użyć następującej wartości specjalnej:

MQCT_NONE

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C jest również zdefiniowana stała MQCT_NONE_ARRAY; ta stała ma taką samą wartość jak MQCT_NONE, ale jest tablicą znaków zamiast łańcucha.

To pole jest używane podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS . W innych środowiskach należy podać wartość MQCT_NONE.

Długość tego pola jest podana przez wartość MQ_CONN_TAG_LENGTH. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_3.

Opcje (MQLONG)

Opcje, które sterują działaniem MQCONN.

Opcje rozliczania

Następujące opcje sterują typem rozliczania, jeśli atrybut menedżera kolejek produktu *AccountingConnOverride* jest ustawiony na wartość MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED

Po wyłączeniu funkcji monitorowania gromadzenia danych w definicji menedżera kolejek przez ustawienie atrybutu *MQIAccounting* na wartość MQMON_OFF, ustawienie tej opcji włącza gromadzenie danych rozliczeniowych MQI.

MQCNO_ACCOUNTING_MQI_DISABLED

Po wyłączeniu funkcji monitorowania gromadzenia danych w definicji menedżera kolejek przez ustawienie atrybutu *MQIAccounting* na wartość MQMON_OFF, ustawienie tej flagi spowoduje zatrzymanie gromadzenia danych rozliczeniowych MQI.

MQCNO_ACCOUNTING_Q_ENABLED

Gdy gromadzenie danych rozliczanych w kolejce jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu *MQIAccounting* na wartość MQMON_OFF, ustawienie tej opcji włącza gromadzenie danych rozliczeniowych dla tych kolejek, które określają menedżer kolejek w polu *MQIAccounting* definicji kolejki.

MQCNO_ACCOUNTING_Q_DISABLED

Gdy gromadzenie danych rozliczanych w kolejce jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu *MQIAccounting* na wartość MQMON_OFF, ustawienie tej opcji wyłącza kolekcjonowanie danych rozliczeniowych dla tych kolejek, które określają menedżer kolejek w polu *MQIAccounting* definicji kolejki.

Jeśli żadna z tych opcji nie jest zdefiniowana, rozliczanie dla połączenia jest zdefiniowane w atrybutach menedżera kolejek.

Opcje powiązania

Następujące opcje sterują typem powiązania IBM MQ, które ma być używane. Określ tylko jedną z następujących opcji:

MQCNO_STANDARD_BINDING

Aplikacja i agent lokalnego menedżera kolejek (komponent zarządzający operacjami kolejkowania) są uruchamiane w oddzielnych jednostkach wykonywania (zwykle w oddzielnych procesach). Ta umowa zachowuje integralność menedżera kolejek, to znaczy zabezpiecza menedżer kolejek przed programami błędnymi.

Jeśli menedżer kolejek obsługuje wiele typów powiązań, a parametr MQCNO_STANDARD_BINDING zostanie ustawiony, menedżer kolejek użyje atrybutu *DefaultBindType* w sekcji *Connection* w pliku *qm.ini*, aby wybrać rzeczywisty typ powiązania. Jeśli ta sekcja nie jest zdefiniowana lub wartość nie może być użyta lub nie jest odpowiednia dla aplikacji, menedżer kolejek wybiera odpowiedni typ powiązania. Menedżer kolejek ustawia rzeczywisty typ powiązania używany w opcjach połączenia.

Użyj opcji MQCNO_STANDARD_BINDING w sytuacjach, w których aplikacja mogła nie zostać w pełni przetestowana lub być może nie jest wiarygodna lub może być niezetelna. Wartość MQCNO_STANDARD_BINDING jest wartością domyślną.

Ta opcja jest obsługiwana we wszystkich środowiskach.

W przypadku łączenia z biblioteką mqm najpierw podejmowana jest próba połączenia standardowego z serwerem przy użyciu domyślnego typu powiązania. Jeśli nie powiodła się próba załadowania bazowej biblioteki serwera, zostanie podjęta próba nawiązania połączenia z klientem.

- Jeśli określono zmienną środowiskową MQ_CONNECT_TYPE, można podać jedną z następujących opcji, aby zmienić zachowanie wartości MQCONN lub MQCONNX, jeśli określono wartość MQCNO_STANDARD_BINDING. (Wyjątkiem jest to, że parametr MQCNO_FASTPATH_BINDING został określony z parametrem MQ_CONNECT_TYPE ustawionym na wartość LOCAL lub STANDARD, aby zezwolić na obniżenie wartości połączeń fastpath przez administratora bez powiązanej zmiany z aplikacją:

Wartość	Znaczenie
KLIENT	Próba nawiązania połączenia z klientem jest wykonywana tylko przez klienta.
Krótką ścieżką	Ta wartość była obsługiwana w poprzednich wersjach, ale została zignorowana, jeśli została określona.
LOKALNA	Próba nawiązania połączenia z serwerem jest wykonywana tylko przez serwer. Połączenia krótkiej ścieżki są obniżane do standardowego połączenia z serwerem.
STANDARDOWA	Obsługiwane w celu zapewnienia zgodności z poprzednimi wersjami. Ta wartość jest teraz traktowana jako LOCAL.

- Jeśli zmienna środowiskowa MQ_CONNECT_TYPE nie jest ustawiona, gdy wywołano wywołanie MQCONNX, podejmowana jest próba standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Jeśli ładowanie biblioteki serwera nie powiedzie się, zostanie podjęta próba nawiązania połączenia z klientem.

MQCNO_FASTPATH_BINDING

Aplikacja i agent lokalnego menedżera kolejek są częścią tej samej jednostki wykonywania. Jest to w przeciwieństwie do typowej metody powiązania, w której aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania.

Wartość MQCNO_FASTPATH_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane, ponieważ nie określono opcji.

Opcja MQCNO_FASTPATH_BINDING może mieć przewagę w sytuacjach, w których wiele procesów zużywa więcej zasobów niż ogólny zasób używany przez aplikację. Aplikacja, która używa powiązania krótkiej ścieżki, jest znana jako *zauwana aplikacja*.

Podczas podejmowania decyzji o użyciu powiązania krótkiej ścieżki należy wziąć pod uwagę następujące ważne punkty:

- Użycie opcji MQCNO_FASTPATH_BINDING nie zapobiega modyfikowaniu lub uszkodzeniu komunikatów oraz innych obszarów danych należących do menedżera kolejek. Tej opcji należy używać tylko w sytuacjach, w których w pełni oceniono te problemy.
- Aplikacja nie może używać sygnałów asynchronicznych ani przerwań licznika czasu (takich jak sigkill) z opcją MQCNO_FASTPATH_BINDING. Istnieją również ograniczenia dotyczące korzystania z segmentów pamięci współużytkowanej.
- Aplikacja musi używać wywołania MQDISC do rozłączenia się z menedżerem kolejek.
- Aplikacja musi zakończyć się przed zakończeniem menedżera kolejek za pomocą komendy endmqm.
- W systemie IBM izadanie musi być uruchamiane w ramach profilu użytkownika należącego do grupy QMQMADM. Ponadto, program nie może się zatrzymać nieprawidłowo, w przeciwnym razie mogą wystąpić nieprzewidywalne rezultaty.
- W systemie Systemy UNIX identyfikator użytkownika mqm musi być efektywnym identyfikatorem użytkownika, a identyfikator grupy mqm musi być efektywnym identyfikatorem grupy. Aby aplikacja została uruchomiona w ten sposób, należy skonfigurować program w taki sposób, aby był on własnością identyfikatora użytkownika produktu mqm i identyfikatora grupy mqm, a następnie ustawić bity uprawnień setuid i setgid w programie.

Program IBM MQ Object Authority Manager (OAM) nadal używa rzeczywistego ID użytkownika do sprawdzania uprawnień.

- W systemie Windows program musi należeć do grupy mqm. Powiązanie krótkiej ścieżki nie jest obsługiwane dla aplikacji 64-bitowych.

Opcja MQCNO_FASTPATH_BINDING jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux i Windows.

 W systemie z/OS opcja jest akceptowana, ale ignorowana.

Więcej informacji na temat implikacji korzystania z zaufanych aplikacji zawiera sekcja [Ograniczenia dla zaufanych aplikacji](#).

MQCNO_SHARED_BINDING

W przypadku komendy MQCNO_SHARED_BINDING aplikacja i agent lokalnego menedżera kolejek współużytkują niektóre zasoby. Wartość MQCNO_SHARED_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

MQCNO_ISOLATED_BINDING

W takim przypadku proces aplikacji i agent lokalnego menedżera kolejek są odizolowane od siebie, ponieważ nie współużytkują zasobów. Wartość MQCNO_ISOLATED_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

MQCNO_CLIENT_BINDING

Tę opcję należy określić, aby aplikacja próbowała wykonać próbę nawiązania połączenia z klientem. Ta opcja ma następujące ograniczenia:

-  Powiązanie MQCNO_CLIENT_BINDING jest ignorowane w systemie z/OS.

- Wartość MQCNO_CLIENT_BINDING jest odrzucana za pomocą wywołania MQRC_OPTIONS_ERROR, jeśli jest ona określona z dowolną opcją powiązania MQCNO inną niż MQCNO_STANDARD_BINDING.
- Powiązanie MQCNO_CLIENT_BINDING nie jest dostępne dla produktu Java lub .NET, ponieważ mają własne mechanizmy wyboru typu powiązania.

MQCNO_LOCAL_BINDING

Tę opcję należy określić, aby aplikacja próbowała nawiązać połączenie z serwerem. Jeśli określono również parametr MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING lub MQCNO_SHARED_BINDING, to połączenie jest typu tego typu i jest udokumentowane w tej sekcji. W przeciwnym razie zostanie podjęta próba użycia standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Parametr MQCNO_LOCAL_BINDING ma następujące ograniczenia:

-  Wartość MQCNO_LOCAL_BINDING jest ignorowana w systemie z/OS.
- Wartość MQCNO_LOCAL_BINDING jest odrzucana za pomocą wywołania MQRC_OPTIONS_ERROR, jeśli jest ona określona z dowolną opcją ponownego połączenia MQCNO, inną niż MQCNO_RECONNECT_AS_DEF.
- Opcja MQCNO_LOCAL_BINDING nie jest dostępna dla produktu Java lub .NET, ponieważ mają własne mechanizmy wyboru typu powiązania.

W systemach AIX, HP-UX, Solaris, Linux i Windows można użyć zmiennej środowiskowej MQ_CONNECT_TYPE z typem powiązania określonym w polu *Options*, aby kontrolować typ używanego powiązania. Jeśli ta zmienna środowiskowa zostanie określona, musi mieć wartość FASTPATH lub STANDARD. Jeśli ma inną wartość, jest ignorowana. W przypadku wartości zmiennej środowiskowej rozróżniana jest wielkość liter; więcej informacji na ten temat zawiera sekcja [Zmienna środowiskowa MQCONN](#).

Zmienna środowiskowa i pole *Options* wchodzi w interakcje w następujący sposób:

- Jeśli zmienna środowiskowa zostanie pominięta lub zostanie podana wartość, która nie jest obsługiwana, użycie powiązania krótkiej ścieżki jest określone wyłącznie przez pole *Options*.
- Jeśli zostanie podana wartość obsługiwana przez zmienną środowiskową, powiązanie krótkiej ścieżki będzie używane tylko wtedy, gdy *obie* zmienne środowiskowe i *Options* określają powiązanie krótkiej ścieżki.

Opcje znaczników połączeń



Te opcje są obsługiwane tylko wtedy, gdy łączy się z menedżerem kolejek produktu z/OS i sterują użyciem znacznika połączenia *ConnTag*. Można określić tylko jedną z następujących opcji:

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Ta opcja żąda wyłączonego użycia znacznika połączenia w lokalnym menedżerze kolejek. Jeśli znacznik połączenia jest już używany w lokalnym menedżerze kolejek, wywołanie MQCONN nie powiedzie się i zostanie użyty kod przyczyny MQRC_CONN_TAG_IN_USE. Na wynik wywołania nie ma wpływu użycie znacznika połączenia w innym miejscu w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

MQCNO_SERIALIZE_CONN_TAG_QSG

Ta opcja żąda wyłączonego użycia znacznika połączenia w grupie współużytkującej kolejkę, do której należy lokalny menedżer kolejek. Jeśli znacznik połączenia jest już używany w grupie współużytkowania kolejki, wywołanie MQCONN nie powiedzie się i zostanie użyty kod przyczyny MQRC_CONN_TAG_IN_USE.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Ta opcja żąda współużytkowanego użycia znacznika połączenia w obrębie lokalnego menedżera kolejek. Jeśli znacznik połączenia jest już używany w lokalnym menedżerze kolejek, wywołanie

MQCONN może zakończyć się powodzeniem, jeśli aplikacja żądająca jest uruchomiona w tym samym zasięgu przetwarzania, co istniejący użytkownik znacznika. Jeśli ten warunek nie zostanie spełniony, wywołanie MQCONN nie powiedzie się i zostanie użyty kod przyczyny MQRC_CONN_TAG_IN_USE. Wynik wywołania nie ma wpływu na użycie znacznika połączenia w innym miejscu w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

- Aby można było współużytkować znacznik połączenia, aplikacje muszą działać w obrębie tego samego przestrzeni adresowej MVS. Jeśli aplikacja używała znacznika połączenia jest aplikacją kliencką, parametr MQCNO_RESTRICT_CONN_TAG_Q_MGR nie jest dozwolony.

MQCNO_RESTRICT_CONN_TAG_QSG

Ta opcja żąda współużytkowanego użycia znacznika połączenia w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek. Jeśli znacznik połączenia jest już używany w grupie współużytkowania kolejki, wywołanie MQCONN może zakończyć się powodzeniem, pod warunkiem że aplikacja żądająca działa w tym samym zasięgu przetwarzania i jest połączona z tym samym menedżerem kolejek, co istniejący użytkownik znacznika.

Jeśli te warunki nie są spełnione, wywołanie MQCONN nie powiedzie się z kodem przyczyny MQRC_CONN_TAG_IN_USE.

- Aby można było współużytkować znacznik połączenia, aplikacje muszą działać w obrębie tego samego przestrzeni adresowej MVS. Jeśli aplikacja używała znacznika połączenia jest aplikacją kliencką, parametr MQCNO_RESTRICT_CONN_TAG_QSG nie jest dozwolony.

Jeśli żadna z tych opcji nie zostanie podana, produkt *ConnTag* nie zostanie użyty. Te opcje nie są poprawne, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_3.

Opcje współużytkowania uchwytu

Opcje te są obsługiwane w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux i Windows. Kontrolują one współużytkowanie uchwytów między różnymi wątkami (jednostkami przetwarzania równoległego) w ramach tego samego procesu. Można określić tylko jedną z następujących opcji:

MQCNO_HANDLE_SHARE_NONE

Ta opcja wskazuje, że uchwytów połączeń i obiektów mogą być używane tylko przez wątek, który spowodował przydzielanie uchwytu (czyli wątku, który wywołał wywołanie MQCONN, MQCONN lub MQOPEN). Uchwytów nie mogą być używane przez inne wątki należące do tego samego procesu.

MQCNO_HANDLE_SHARE_BLOCK

Ta opcja wskazuje, że uchwytów połączeń i obiektów przydzielone przez jeden wątek procesu mogą być używane przez inne wątki należące do tego samego procesu. Jednak tylko jeden wątek w danym momencie może użyć dowolnego uchwytu, to znaczy, że dozwolone jest tylko użycie numeru seryjnego uchwytu. Jeśli wątek próbuje użyć uchwytu, który jest już używany przez inny wątek, bloki wywołań (czeka), aż do momentu, gdy uchwyt stanie się dostępny.

MQCNO_HANDLE_SHARE_NO_BLOCK

Jest to taka sama sytuacja, jak w przypadku komendy MQCNO_HANDLE_SHARE_BLOCK, z tą różnicą, że jeśli uchwyt jest używany przez inny wątek, wywołanie zakończy się natychmiast po zakończeniu operacji MQCC_FAILED i MQRC_CALL_IN_PROGRESS, a nie do momentu, gdy uchwyt stanie się dostępny.

Wątek może mieć zero lub jeden niewspółużytkowany uchwyt:

- Każde wywołanie MQCONN lub MQCONN, które określa parametr MQCNO_HANDLE_SHARE_NONE, zwraca nowy niewspółużytkowany uchwyt w pierwszym wywołaniu i ten sam niewspółużytkowany uchwyt w przypadku wywołań drugiego i późniejszego (przy założeniu, że nie można było wywołać wywołania MQDISC). Kod przyczyny to MQRC_ALREADY_CONNECTED w przypadku wywołań drugiego i późniejszego.

- Każde wywołanie MQCONN, które określa parametr MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwraca nowy współużytkowany uchwyt dla każdego wywołania.

Uchwyty obiektów dziedziczą te same właściwości współużytkowania, co uchwyt połączenia określony w wywołaniu MQOPEN, które utworzyło uchwyt obiektu. Ponadto jednostki pracy dziedziczą te same właściwości współużytkowania, co uchwyt połączenia używany do uruchamiania jednostki pracy. Jeśli jednostka pracy jest uruchamiana w jednym wątku przy użyciu współużytkowanego uchwyty, to jednostka pracy może być aktualizowana w innym wątku przy użyciu tego samego uchwyty.

Jeśli opcja współużytkowania uchwyty nie zostanie określona, wartość domyślna jest określana przez środowisko:

- W środowisku produktu Microsoft Transaction Server (MTS) wartość domyślna jest taka sama jak wartość MQCNO_HANDLE_SHARE_BLOCK.
- W innych środowiskach wartość domyślna jest taka sama jak wartość MQCNO_HANDLE_SHARE_NONE.

Opcje ponownego połączenia

Opcje ponownego połączenia określają, czy połączenie jest nawiązane ponownie. Tylko połączenia klienckie są ponownie nawiązane.

MQCNO_RECONNECT_AS_DEF

Opcja ponownego połączenia jest tłumaczana na wartość domyślną. Jeśli wartość domyślna nie jest ustawiona, wartość tej opcji jest tłumaczona na DISABLED. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

MQCNO_RECONNECT,

Aplikacja może zostać ponownie połączona z dowolnym menedżerem kolejek zgodnym z wartością parametru QmgrName produktu MQCONN. Opcji MQCNO_RECONNECT należy używać tylko wtedy, gdy nie ma powinowactwa między aplikacją kliencką a menedżerem kolejek, z którym początkowo nawiązało połączenie. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

MQCNO_RECONNECT_DISABLED

Nie można ponownie nawiązać połączenia z aplikacją. Wartość opcji nie jest przekazywana do serwera.

MQCNO_RECONNECT_Q_MGR

Aplikacja może zostać ponownie połączona tylko z menedżerem kolejek, z którym pierwotnie był połączony. Tej wartości należy użyć, jeśli klient może być ponownie połączony, ale istnieje powinowactwo między aplikacją kliencką a menedżerem kolejek, z którym pierwotnie nawiązało połączenie. Wartość tę należy wybrać, jeśli klient ma automatycznie nawiązywać ponowne połączenie z instancją rezerwową menedżera kolejek o wysokiej dostępności. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

Należy użyć opcji MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED i MQCNO_RECONNECT_Q_MGR tylko dla połączeń klienckich. Jeśli opcje są używane dla połączenia powiązania, MQCONN nie powiedzie się z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_OPTIONS_ERROR. Automatyczne ponowne łączenie klienta nie jest obsługiwane przez produkt IBM MQ classes for Java

Opcje współużytkowania konwersacji

Poniższe opcje mają zastosowanie tylko do połączeń klientów TCP/IP. W przypadku kanałów SNA, SPX i NetBios wartości te są ignorowane, a kanał działa tak jak w poprzednich wersjach produktu.

MQCNO_NO_CONV_SHARING

Ta opcja nie zezwala na współużytkowanie konwersacji.

Opcji MQCNO_NO_CONV_SHARING można użyć w sytuacjach, w których konwersacje są mocno obciążone, a w związku z tym, w przypadku, gdy rywalizacja jest możliwością na zakończenie połączenia z serwerem instancji kanału, na którym istnieją konwersacje współużytkowania. Funkcja MQCNO_NO_CONV_SHARING zachowuje się, jak sharecnv (1), gdy jest podłączony do kanału, który obsługuje współużytkowanie konwersacji, i sharecnv (0), gdy jest podłączony do kanału, który nie obsługuje współużytkowania konwersacji.

MQCNO_ALL_CONVS_SHARE

Ta opcja umożliwia współużytkowanie konwersacji. Aplikacja nie ma żadnych ograniczeń dotyczących liczby połączeń w instancji kanału. Ta opcja jest wartością domyślną.

Jeśli aplikacja wskazuje, że instancja kanału może współużytkować, ale definicja *SharingConversations* (SHARECNV) na końcu połączenia z serwerem jest ustawiona na wartość 1, współużytkowanie nie jest wykonywane i żadne ostrzeżenie nie jest wyświetlane dla aplikacji.

Podobnie, jeśli aplikacja wskazuje, że współużytkowanie jest dozwolone, ale definicja *SharingConversations* połączenia z serwerem jest ustawiona na zero, ostrzeżenie nie jest wyświetlane, a aplikacja wykazuje takie samo zachowanie, jak klient w wersjach produktu wcześniejszych niż produkt Wersja 7.0. Ustawienie aplikacji związane z współużytkowaniem konwersacji jest ignorowane.

Opcja MQCNO_NO_CONV_SHARING i MQCNO_ALL_CONVS_SHARE wzajemnie się wykluczają. Jeśli obie opcje są określone dla określonego połączenia, połączenie zostanie odrzucone z kodem przyczyny MQRC_OPTIONS_ERROR.

Opcje definicji kanału

Następujące opcje sterują użyciem struktury definicji kanału przekazanej w MQCNO:

MQCNO_CD_FOR_OUTPUT_OUTPUT_ONLY

Ta opcja umożliwia użycie struktury definicji kanału w strukturze MQCNO tylko w celu zwrócenia nazwy kanału używanej w pomyślnym wywołaniu MQCONN.

Jeśli nie zostanie podana poprawna struktura definicji kanału, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_CD_ERROR.

Jeśli aplikacja nie jest uruchomiona jako klient, opcja jest ignorowana.

Zwrócona nazwa kanału może zostać użyta podczas kolejnego wywołania MQCONN przy użyciu opcji MQCNO_USE_CD_SELECTION w celu ponownego nawiązania połączenia przy użyciu tej samej definicji kanału. Może to być przydatne w sytuacji, gdy w tabeli kanału klienta istnieje wiele odpowiednich definicji kanałów.

MQCNO_USE_CD_SELECTION

Ta opcja umożliwia wywołanie komendy MQCONN w celu nawiązania połączenia przy użyciu nazwy kanału zawartej w strukturze definicji kanału przekazanej w MQCNO.

Jeśli ustawiona jest zmienna środowiskowa MQSERVER, używana jest definicja kanału zdefiniowana przez tę zmienną. Jeśli wartość MQSERVER nie jest ustawiona, używana jest tabela kanałów klienta.

Jeśli definicja kanału o zgodnej nazwie kanału i nazwie menedżera kolejek nie zostanie znaleziona, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_Q_MGR_NAME_ERROR.

Jeśli nie zostanie podana poprawna struktura definicji kanału, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_CD_ERROR.

Jeśli aplikacja nie jest uruchomiona jako klient, opcja jest ignorowana.

Opcja domyślna

Jeśli nie jest wymagana żadna z opisanych powyżej opcji, można użyć następującej opcji:

MQCNO_NONE

Nie określono żadnych opcji.

Do dokumentacji programu pomocy należy użyć komendy MQCNO_NONE. Ta opcja nie jest używana z żadną inną opcją MQCNO_*, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

SecurityParmsPrzesunięcie (MQLONG)

SecurityParmsPrzesunięcie jest to przesunięcie w bajtach struktury MQCSP od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym, którego początkowa wartość wynosi 0.

To pole jest ignorowane, jeśli *wersja* jest mniejsza niż MQCNO_VERSION_5.

Struktura MQCSP jest zdefiniowana w produkcie [“MQCSP-parametry zabezpieczeń”](#) na stronie 336.

SecurityParmsPtr (PMQCSP)

SecurityParmsPtr jest adresem struktury MQCSP, używanym do określania identyfikatora użytkownika i hasła na potrzeby uwierzytelniania przez usługę autoryzacji. To pole jest polem wejściowym, a jego wartością początkową jest pusty wskaźnik lub null bajtów.

To pole jest ignorowane, jeśli *wersja* jest mniejsza niż MQCNO_VERSION_5.

Struktura MQCSP jest zdefiniowana w produkcie [“MQCSP-parametry zabezpieczeń”](#) na stronie 336.

SSLConfigOffset (MQLONG)

SSLConfigOffset to przesunięcie w bajtach struktury MQSCO od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym, którego początkowa wartość wynosi 0.

Opcji *SSLConfigOffset* należy używać tylko wtedy, gdy aplikacja wywołujące wywołanie MQCONN jest uruchomiona jako IBM MQ MQI client. Informacje na temat korzystania z tego pola można znaleźć w opisie pola *SSLConfigPtr*.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_4.

SSLConfigPtr (PMQSCO)

SSLConfigPtr jest polem wejściowym. Jego wartością początkową jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Opcji *SSLConfigPtr* i *SSLConfigOffset* należy używać tylko wtedy, gdy aplikacja wywołujące wywołanie MQCONN jest uruchomiona jako IBM MQ MQI client, a protokołem kanału jest protokół TCP/IP. Jeśli aplikacja nie jest uruchomiona jako klient IBM MQ lub protokół kanału nie jest protokołem TCP/IP, to *SSLConfigPtr* i *SSLConfigOffset* są ignorowane.

Podanie wartości *SSLConfigPtr* lub *SSLConfigOffset* plus *ClientConnPtr* lub *ClientConnOffset* powoduje, że aplikacja może sterować używaniem protokołu SSL dla połączenia klienta. Jeśli informacje SSL zostaną podane w ten sposób, zmienne środowiskowe MQSSLKEYR i MQSSLCRYP są ignorowane; wszystkie informacje związane z protokołem SSL w tabeli definicji kanału klienta (CCDT) są również ignorowane.

Informacje o protokole SSL mogą być określone tylko w następujących:

- Pierwsze wywołanie MQCONN procesu klienta, lub
- Kolejne wywołanie MQCONN, gdy wszystkie poprzednie połączenia SSL/TLS z menedżerem kolejek zostały zakończone za pomocą MQDISC.

Są to jedyne stany, w których możliwe jest zainicjowanie środowiska SSL w całym procesie. Jeśli wywołanie MQCONN jest wysyłane, podając informacje SSL, gdy środowisko SSL już istnieje, informacje SSL w wywołaniu są ignorowane, a połączenie jest nawiązywana za pomocą istniejącego

środowiska SSL; w tym przypadku wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SSL_ALREADY_ZAINICJOWANY.

Strukturę MQSCO można udostępnić w taki sam sposób, jak struktura MQCD, podając adres w programie *SSLConfigPtr* lub określając przesunięcie w składce *SSLConfigOffset*. zobacz opis produktu *ClientConnPtr*, aby uzyskać szczegółowe informacje na temat tego, jak to zrobić. Można jednak używać nie więcej niż jednej z następujących produktów: *SSLConfigPtr* i *SSLConfigOffset*; Wywołanie nie powiodło się. Kod przyczyny: MQRC_SSL_CONFIG_ERROR. Jeśli oba są niezerowe.

Po zakończeniu wywołania MQCONN, struktura MQSCO nie jest przywoływana ponownie.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_4.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

StrucId (MQCHAR4)

StrucId jest zawsze polem wejściowym. Jego początkowa wartość to MQCNO_STRUC_ID.

Wartość musi być następująca:

MQCNO_STRUC_ID

Identyfikator struktury opcji łączenia.

Dla języka programowania C jest również zdefiniowana stała zmienna MQCNO_STRUC_ID_ARRAY; ta stała ma taką samą wartość jak MQCNO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG)

Wersja jest zawsze polem wejściowym. Jego początkowa wartość to MQCNO_VERSION_1.

Wartość musi być jedną z następujących wartości:

MQCNO_VERSION_1

Struktura opcji łączenia Version-1.

MQCNO_VERSION_2

Struktura opcji łączenia Version-2.

MQCNO_VERSION_3

Struktura opcji connect-options Version-3.

MQCNO_VERSION_4

Struktura opcji łączenia Version-4.

MQCNO_VERSION_5

Struktura opcji connect-options Version-5.

Ta wersja struktury MQCNO rozszerza MQCNO_VERSION_3 w systemach z/OSi MQCNO_VERSION_4 na wszystkich innych platformach.

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MQCNO_CURRENT_VERSION

Bieżąca wersja struktury opcji połączenia.

Wartości początkowe i deklaracje języków dla MQCNO

Tabela 28. Początkowe wartości pól w MQCNO dla MQCNO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQCNO_STRUC_ID	'CNO~'
<i>Version</i>	MQCNO_VERSION_1	1
<i>Options</i>	MQCNO_NONE	0
<i>ClientConnOffset</i>	Brak	0

Tabela 28. Początkowe wartości pól w MQCNO dla MQCNO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>ClientConnPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>ConnTag</i>	MQCT_NONE	Wartości null
<i>SSLConfigPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>SSLConfigOffset</i>	Brak	0
<i>ConnectionId</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>SecurityParmsOffset</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>SecurityParmsPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraParametr MQCNO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
    MQCHAR4    StructId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     Options;           /* Options that control the action of
    MQCONNXX */
    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 */
};
```

Deklaracja języka COBOL

```
** MQCNO structure
10 MQCNO.
** Structure identifier
15 MQCNO-STRUCID PIC X(4).
** Structure version number
15 MQCNO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQCONNXX
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.

```

Deklaracja PL/I

```

dcl
1 MQCNO based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 Options          fixed bin(31),    /* Options that control the action
                                     of 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 */

```

Deklaracja High Level Assembler

MQCNO	DSECT		
MQCNO_STRUCID	DS	CL4	Structure identifier
MQCNO_VERSION	DS	F	Structure version number
MQCNO_OPTIONS	DS	F	Options that control the action of
*			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	
	ORG	MQCNO	
MQCNO_AREA	DS	CL(MQCNO_LENGTH)	

Wizualna deklaracja podstawowa

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-parametry zabezpieczeń

W poniższej tabeli podsumowano pola w strukturze.

Ostrzeżenie: W niektórych przypadkach hasło w strukturze MQCSP dla aplikacji klienckiej zostanie wysłane przez sieć w postaci jawnego tekstu. Aby upewnić się, że hasła aplikacji klienta są odpowiednio chronione, należy zapoznać się z [IBM MQochroną hasła CSP](#).

Tabela 29. Pola w tabeli MQCSP		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>AuthenticationType</i>	Typ uwierzytelniania	AuthenticationType
<i>Reserved1</i>	Wymagane do wyrównania wskaźnika w systemie IBM i	Reserved1
<i>CSPUserIdPtr</i>	Adres ID użytkownika	CSPUserIdPtr
<i>CSPUserIdOffset</i>	Przesunięcie ID użytkownika	CSPUserIdPrzesunięcie
<i>CSPUserIdLength</i>	Długość ID użytkownika	CSPUserIdDługość
<i>Reserved2</i>	Wymagane do wyrównania wskaźnika w systemie IBM i	Reserved2
<i>CSPPasswordPtr</i>	Adres hasła	CSPPasswordPtr
<i>CSPPasswordOffset</i>	Przesunięcie hasła	CSPPasswordOffset
<i>CSPPasswordLength</i>	Długość hasła	CSPPasswordLength

Przegląd protokołu MQCSP

Dostępność: Wszystkie produkty IBM MQ .

Cel: Struktura MQCSP umożliwia usłudze autoryzacji uwierzytelnianie identyfikatora użytkownika i hasła. Strukturę parametrów zabezpieczeń połączenia MQCSP określa się w wywołaniu MQCONN.

Ostrzeżenie: W niektórych przypadkach hasło w strukturze MQCSP dla aplikacji klienckiej zostanie wysłane przez sieć w postaci jawnego tekstu. Aby upewnić się, że hasła aplikacji klienta są odpowiednio chronione, należy zapoznać się z [IBM MQochroną hasła CSP](#).

Zestaw znaków i kodowanie: Dane w produkcie MQCSP muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Te dane są nadawane odpowiednio przez atrybut menedżera kolejek produktu *CodedCharSetId* i atrybut MQENC_NATIVE.

Pola dla protokołu MQCSP

Struktura MQCSP zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

AuthenticationType (MQLONG)

AuthenticationType to pole wejściowe. Jego początkowa wartość to MQCSP_AUTH_NONE.

Jest to typ uwierzytelniania do wykonania. Poprawne wartości:

MQCSP_AUTH_NONE

Nie należy używać pól identyfikatora użytkownika i hasła.

MQCSP_AUTH_USER_ID_AND_PWD

Uwierzytelniaj pola ID użytkownika i hasła.

Wartością domyślną jest MQCSP_AUTH_NONE. Ustawienie domyślne oznacza, że ochrona hasłem nie jest wykonywana.

Jeśli wymagane jest uwierzytelnianie, należy ustawić wartość **MQCSP.AuthenticationType** do MQCSP_AUTH_USER_ID_AND_PWD.

Więcej informacji na ten temat zawiera sekcja [Zabezpieczenie hasłem protokołu MQCSP](#).

CSPPasswordLength (MQLONG)

To pole jest długością hasła, które ma być używane w uwierzytelnianiu.

Maksymalna długość hasła to MQ_CSP_PASSWORD_LENGTH. Jeśli długość hasła jest większa niż maksymalna dozwolona długość hasła, żądanie uwierzytelnienia nie powiedzie się i zostanie wysłane żądanie MQRC_NOT_AUTHORIZED.

Wartość MQ_CSP_PASSWORD_LENGTH wynosi 256.

To pole jest polem wejściowym. Wartością początkową tego pola jest 0.

CSPPasswordOffset (MQLONG)

Jest to przesunięcie w bajtach hasła, które ma być używane w uwierzytelnianiu. Przesunięcie może być dodatnie lub ujemne.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSPPasswordPtr (MQPTR)

Jest to adres (w bajtach) hasła, który ma być używany w uwierzytelnianiu.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_5.

To pole może zawierać puste hasło, które jest odrzucane przez system operacyjny lub sprawdzanie hasła LDAP, w zależności od konfiguracji, ale nie jest odrzucane przez produkt IBM MQ przed przekazaniem go do metody uwierzytelniania.

Długość CSPUserId (MQLONG)

To pole jest długością identyfikatora użytkownika, który ma być używany w uwierzytelnianiu.

Maksymalna długość identyfikatora użytkownika zależy od platformy. Patrz sekcja [Identyfikatory użytkowników](#). Jeśli długość identyfikatora użytkownika jest większa niż maksymalna dozwolona długość, żądanie uwierzytelniania kończy się niepowodzeniem z opcją MQRC_NOT_AUTHORIZED.

To pole jest polem wejściowym. Wartością początkową tego pola jest 0.

CSPUserIdPrzesunięcie (MQLONG)

Jest to przesunięcie w bajtach identyfikatora użytkownika, który ma być używany w uwierzytelnianiu. Przesunięcie może być dodatnie lub ujemne.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSPUserIdPtr (MQPTR)

Jest to adres (w bajtach) identyfikatora użytkownika, który ma być używany do uwierzytelniania.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_5.

To pole może zawierać identyfikator użytkownika systemu operacyjnego, jeśli w polu CONNAUTH menedżera kolejek znajduje się nazwa **AUTHTYPE** systemu *IDPWOS*.

W systemie Windows może to być pełny identyfikator użytkownika domeny.

To pole może zawierać identyfikator użytkownika LDAP, jeśli w polu KONNAUTH menedżera kolejek znajduje się nazwa **AUTHTYPE** produktu *IDPWLDAP*.

Reserved1 (MQBYTE4)

Pole zarezerwowane, wymagane do wyrównania wskaźnika w systemie IBM i.

To jest pole wejściowe. Początkowa wartość tego pola jest równa null.

Reserved2 (MQBYTE8)

Pole zarezerwowane, wymagane do wyrównania wskaźnika w systemie IBM i.

To jest pole wejściowe. Początkowa wartość tego pola jest równa null.

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość musi być następująca:

MQCSP_STRUC_ID,

Identyfikator struktury parametrów zabezpieczeń.

Dla języka programowania C zdefiniowana jest również stała MQCSP_STRUC_ID_ARRAY; ma taką samą wartość jak MQCSP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCSPSTRUC_ID.

Wersja (MQLONG)

Numer wersji struktury.

Wartość musi być następująca:

MQCSP_VERSION_1

Struktura parametrów zabezpieczeń Version-1.

Następująca stała określa numer wersji bieżącej wersji:

MQCSP_CURRENT_VERSION

Bieżąca wersja struktury parametrów zabezpieczeń.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCSP_VERSION_1.

Wartości początkowe i deklaracje języka dla protokołu MQCSP

Tabela 30. Początkowe wartości pól w tabeli MQCSP dla protokołu MQCSP		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQCSP_STRUC_ID,	'CSP-'
<i>Version</i>	MQCSP_VERSION_1	1
<i>AuthenticationType</i>	Brak	MQCSP_AUTH_NONE
<i>Reserved1</i>	Brak	Pusty łańcuch lub odstępy

Tabela 30. Początkowe wartości pól w tabeli MQCSP dla protokołu MQCSP (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
CSPUserIdPtr	Brak	Pusty wskaźnik lub zerowe bajty
CSPUserIdOffset	Brak	0
CSPUserIdLength	Brak	0
Reserved2	Brak	Pusty łańcuch lub odstępy
CSPPasswordPtr	Brak	Pusty wskaźnik lub zerowe bajty
CSPPasswordOffset	Brak	0
CSPPasswordLength	Brak	0

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraParametr MQCSP_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     AuthenticationType; /* Type of authentication */
    MQBYTE4    Reserved1;        /* Required for IBM i pointer
                                alignment */
    MQPTR      CSPUserIdPtr;      /* Address of user ID */
    MQLONG     CSPUserIdOffset;   /* Offset of user ID */
    MQLONG     CSPUserIdLength;   /* Length of user ID */
    MQBYTE8    Reserved2;        /* Required for IBM i pointer
                                alignment */
    MQPTR      CSPPasswordPtr;    /* Address of password */
    MQLONG     CSPPasswordOffset; /* Offset of password */
    MQLONG     CSPPasswordLength; /* Length of password */
};
```

Deklaracja języka COBOL

```
**  MQCSP structure
10  MQCSP.
**  Structure identifier
15  MQCSP-STRUCID      PIC X(4).
**  Structure version number
15  MQCSP-VERSION     PIC S9(9) BINARY.
**  Type of authentication
15  MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**  Required for IBM i pointer alignment
15  MQCSP-RESERVED1    PIC X(4).
**  Address of user ID
15  MQCSP-CSPUSERIDPTR    POINTER.
**  Offset of user ID
15  MQCSP-CSPUSERIDOFFSET PIC S9(9) BINARY.
**  Length of user ID
15  MQCSP-CSPUSERIDLENGTH PIC S9(9) BINARY.
```

```

**      Required for IBM i pointer alignment
15 MQCSP-RESERVED2      PIC X(4).
**      Address of password
15 MQCSP-CSPPASSWORDPTR  POINTER.
**      Offset of password
15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**      Length of password
15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQCSP based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 AuthenticationType fixed bin(31),  /* Type of authentication */
3 Reserved1        char(4),          /* Required for IBM i pointer
                                     alignment */
3 CSPUserIdPtr     pointer,           /* Address of user ID */
3 CSPUserIdOffset  fixed bin(31),    /* Offset of user ID */
3 CSPUserIdLength  fixed bin(31),    /* Length of user ID */
3 Reserved2        char(8),          /* Required for IBM i pointer
                                     alignment */
3 CSPPasswordPtr   pointer,           /* Address of password */
3 CSPPasswordOffset fixed bin(31),   /* Offset of user ID */
3 CSPPasswordLength fixed bin(31);  /* Length of user ID */

```

Wizualna deklaracja podstawowa

```

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-Struktura opcji wywołania zwrotnego elementu sterującego

W poniższej tabeli podsumowano pola w strukturze. Struktura określająca funkcję wywołania zwrotnego elementu sterującego.

Tabela 31. Pola w tabeli MQCTLO		
Pole	Opis	Temat
<i>StrucID</i>	Identyfikator struktury	StrucID
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje
<i>Reserved</i>	Zarezerwowane pole	Opcje
<i>ConnectionArea</i>	Pole dla funkcji zwrotnej do użycia	ConnectionArea

Przegląd produktu MQCTLO

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSi IBM MQ MQI clients połączone z tymi systemami. Przegląd struktury MQCTLO.

Cel: Struktura MQCTLO służy do określania opcji odnoszących się do funkcji zwrotnych sterowania.

Struktura jest parametrem wejściowym i wyjściowym w wywołaniu komendy MQCTL .

Wersja: Bieżąca wersja obiektu MQCTLO to MQCTLO_VERSION_1.

Zestaw znaków i kodowanie: Dane w tabeli MQCTLO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQCTLO

Alfabetyczna lista pól dla struktury MQCTLO.

Struktura MQCTLO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

ConnectionArea (MQPTR),

Struktura opcji elementu sterującego-pole ConnectionArea

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian do pola ConnectionArea w strukturze MQCBC, która jest parametrem wejściowym wywołania zwrotnego.

To pole jest ignorowane dla wszystkich operacji innych niż MQOP_START i MQOP_START_WAIT.

Jest to pole wejściowe i wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

Opcje (MQLONG)

Struktura opcji kontrolnych-pole Opcje

Opcje sterujące działaniem komendy MQCTL.

MQCTLO_FAIL_IF QUIESCING

Wymuś niepowodzenie wywołania MQCTL, jeśli menedżer kolejek lub połączenie znajduje się w stanie wygaszania.

Podaj wartość MQGMO_FAIL_IF QUIESCING, w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadamianie konsumentów komunikatów, gdy są one wygaszane.

MQCTLO_THREAD_AFFINITY

Ta opcja informuje system, że aplikacja wymaga, aby wszystkie konsumenty komunikatów, dla tego samego połączenia, były wywoływane w tym samym wątku. Ten wątek będzie używany dla wszystkich wywołań konsumentów, dopóki połączenie nie zostanie zatrzymane.

Opcja domyślna: Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

MQCTLO_BRAK

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Parametr MQCTLO_NONE jest zdefiniowany w dokumentacji programu pomocowego. Nie jest on przeznaczony do użycia z żadną inną opcją, ale ponieważ jej wartość jest równa zero, tego typu użycie nie może zostać wykryte.

To jest pole wejściowe. Wartością początkową w polu *Options* jest MQCTLO_NONE.

Zarezerwowane (MQLONG)

Jest to pole zastrzeżone. Wartość musi być równa zero.

StrucId (MQCHAR4)

Struktura opcji elementu sterującego-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQCTLO_STRUC_ID

Identyfikator struktury opcji sterowania.

Dla języka programowania w języku C jest również zdefiniowana stała MQCTLO_STRUC_ID_ARRAY; ta sama wartość ma wartość MQCTLO_STRUC_ID, ale jest to tablica znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCTLO_STRUC_ID.

Wersja (MQLONG)

Struktura opcji kontrolnych-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQCTLO_VERSION_1

Struktura opcji sterowania Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQCTLO_CURRENT_VERSION

Bieżąca wersja struktury opcji sterowania.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQCTLO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQCTLO

Struktura opcji kontrolnych-wartości początkowe

Tabela 32. Początkowe wartości pól w MQCTLO		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQCTLO_STRUC_ID	'CTLO'
Version	MQCTLO_VERSION_1	1
Options	MQCTLO_BRAK	Wartości null
Reserved	Zarezerwowane pole	
ConnectionArea	Brak	Pusty wskaźnik lub zerowe bajty
Uwagi: 1. W języku programowania C: zmienna makraParametr MQCTLO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQCTLO MyCTLO = {MQCTLO_DEFAULT};</pre>		

Deklaracja C

Struktura opcji kontroli-deklaracja języka C

```
typedef struct tagMQCTLO MQCTLO;  
struct tagMQCTLO {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;           /* Structure version number */  
    MQLONG     Options;           /* Options that control the action of MQCTL */  
    MQLONG     Reserved;          /* Reserved field */  
  
    MQPTR      ConnectionArea; /* Connection work area passed to the function */  
};
```

Deklaracja języka COBOL

```

** MQCTLO structure
10 MQCTLO.
** Structure Identifier
15 MQCTLO-STRUCID PIC X(4).
** Structure Version
15 MQCTLO-VERSION PIC S9(9) BINARY.
** Options
15 MQCTLO-OPTIONS PIC S9(9) BINARY.
** Reserved
15 MQCTLO-RESERVED PIC S9(9) BINARY.
** ConnectionArea
15 MQCTLO-CONNECTIONAREA POINTER

```

Deklaracja PL/I

```

dcl
1 MQCTLO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version */
3 Options fixed bin(31), /* Options */
3 Reserved fixed bin(31),
3 ConnectionArea pointer; /* Connection work area */

```

MQDH-nagłówek dystrybucji

W poniższej tabeli podsumowano pola w strukturze.

Tabela 33. Pola w MQDH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQDH oraz następujące rekordy	StrucLength
<i>Encoding</i>	Kodowanie numeryczne danych, które są zgodne z tablicą rekordów MQPMR	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków danych, które są zgodne z tablicą rekordów MQPMR	CodedCharSetId
<i>Format</i>	Format nazwy danych, które są zgodne z tablicą rekordów MQPMR	Formatowanie
<i>Flags</i>	Flagi ogólne	Flagi
<i>PutMsgRecFields</i>	Flagi wskazujące, które pola MQPMR są obecne	PutMsgRecFields
<i>RecsPresent</i>	Liczba obecnych rekordów obiektów	RecsPresent
<i>ObjectRecOffset</i>	Przesunięcie pierwszego rekordu obiektu od początku wywołania MQDH	ObjectRecPrzesunięcie
<i>PutMsgRecOffset</i>	Przesunięcie pierwszego rekordu komunikatu umieszczonego od początku wywołania MQDH	PutMsgRecOffset

Przegląd produktu MQDH

Dostępność: klienci AIX, HP-UX, IBM i, Solaris, Linux, Windowsi IBM MQ połączone z tymi systemami.

Cel: Struktura MQDH opisuje dodatkowe dane, które są obecne w komunikacie, gdy ten komunikat jest komunikatem listy dystrybucyjnej przechowywanej w kolejce transmisji. Komunikat z listą dystrybucyjną

jest to komunikat wysyłany do wielu kolejek docelowych. Dodatkowe dane składają się z struktury MQDH, po której następuje tablica rekordów MQOR i tablica rekordów MQPMR.

Ta struktura jest używana przez wyspecjalizowane aplikacje, które umieszczają komunikaty bezpośrednio w kolejkach transmisji lub usuwają komunikaty z kolejek transmisji (na przykład: agenty kanałów komunikatów).

Aplikacje, które mają umieszczać komunikaty w listach dystrybucyjnych, nie mogą używać tej struktury. Zamiast tego muszą one używać struktury MQOD do definiowania miejsc docelowych na liście dystrybucyjnej oraz struktury MQPMO w celu określenia właściwości komunikatu lub otrzymywania informacji o komunikatach wysyłanych do poszczególnych miejsc docelowych.

Nazwa formatu: MQFMT_DIST_HEADER.

Zestaw znaków i kodowanie: Dane w tabeli MQDH muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie wartości MQDH w polach *CodedCharSetId* i *Encoding* w:

- MQMD (jeśli struktura MQDH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQDH (wszystkie inne obserwacje).

Użycie: Gdy aplikacja umieszcza komunikat na liście dystrybucyjnej, a niektóre lub wszystkie miejsca docelowe są zdalne, menedżer kolejek prefikuje dane komunikatu aplikacji za pomocą struktur MQXQH i MQDH, a następnie umieszcza komunikat w odpowiedniej kolejce transmisji. W związku z tym dane są wykonywane w następującej kolejności, gdy komunikat znajduje się w kolejce transmisji:

- Struktura MQXQH
- Struktura MQDH plus tablice rekordów MQOR i MQPMR
- Dane komunikatu aplikacji

W zależności od miejsc docelowych menedżer kolejek może wygenerować więcej niż jeden taki komunikat i umieścić go w różnych kolejkach transmisji. W tym przypadku struktury MQDH w tych komunikatach identyfikują różne podzbiory miejsc docelowych zdefiniowanych przez listę dystrybucyjną otwieraną przez aplikację.

Aplikacja, która umieszcza komunikat z listą dystrybucyjną bezpośrednio w kolejce transmisji, musi być zgodna z opisaną wcześniej sekwencją i musi się upewnić, że struktura MQDH jest poprawna. Jeśli struktura MQDH nie jest poprawna, menedżer kolejek może nie powieść się z wywołania MQPUT lub MQPUT1 z kodem przyczyny MQRC_DH_ERROR.

Komunikaty można przechowywać w kolejce w postaci listy dystrybucyjnej tylko wtedy, gdy kolejka jest zdefiniowana jako możliwa do obsługi komunikatów listy dystrybucyjnej (patrz atrybut kolejki *DistLists* opisany w sekcji „Atrybuty dla kolejek” na stronie 845). Jeśli aplikacja umieszcza komunikat z listą dystrybucyjną bezpośrednio w kolejce, która nie obsługuje listy dystrybucyjnej, menedżer kolejek rozdziela komunikat listy dystrybucyjnej na poszczególne komunikaty i zamiast niego umieszcza je w kolejce.

Pola dla MQDH

Struktura MQDH zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**:

CodedCharSetId (MQLONG)

Jest to identyfikator zestawu znaków danych, który jest zgodny z tablicami rekordów MQOR i MQPMR. Nie ma on zastosowania do danych znakowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych po tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wystanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wywołanie MQGET nie zwraca wartości MQCCSI_INHERIT.

Nie można użyć komendy MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcie MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz klienty IBM MQ połączone z tymi systemami.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Kodowanie (MQLONG)

Jest to kodowanie liczbowe dla danych, które są zgodne z tablicami rekordów MQOR i MQPMR. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

Flagi (MQLONG)

Można podać następującą opcję:

MQDHF_NEW_MSG_IDS

Wygeneruj nowy identyfikator komunikatu dla każdego miejsca docelowego na liście dystrybucyjnej. Tę opcję należy ustawić tylko wtedy, gdy nie ma rekordów put-message, lub gdy rekordy są obecne, ale nie zawierają pola *MsgId*.

Użycie tej opcji umożliwia defenowanie identyfikatorów komunikatów aż do momentu, gdy komunikat listy dystrybucyjnej zostanie ostatecznie podzielony na poszczególne komunikaty. Minimalizuje to ilość informacji sterujących, które muszą przepływać wraz z komunikatem listy dystrybucyjnej.

Gdy aplikacja wstawi komunikat do listy dystrybucyjnej, menedżer kolejek ustawia w MQDH wartość MQDHF_NEW_MSG_IDS, która generuje, gdy spełniony jest oba z następujących warunków:

- Brak rekordów umieszczania komunikatów udostępnionych przez aplikację lub udostępnione rekordy nie zawierają pola *MsgId* (Nie zawiera rekordów zawierających komunikat).
- Pole *MsgId* w strukturze MQMD to MQMI_NONE lub pole *Options* w produkcie MQPMO zawiera wartość MQPMO_NEW_MSG_ID.

Jeśli nie są wymagane żadne opcje, należy podać następujące informacje:

MQDHF_NONE

Nie określono żadnych flag. Wartość MQDHF_NONE jest zdefiniowana w dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest MQDHF_NONE.

Format (MQCHAR8)

Jest to nazwa formatu danych, które są zgodne z tablicami rekordów MQOD i MQPMR (w zależności od tego, co nastąpi ostatnio).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

ObjectRecPrzesunięcie (MQLONG)

Daje to przesunięcie w bajtach pierwszego rekordu w tablicy rekordów obiektów MQOR, zawierających nazwy kolejek docelowych. W tej tablicy znajdują się rekordy *RecsPresent*. Rekordy te (plus wszystkie bajty pominięte między pierwszym rekordem obiektu a poprzednim polem) są uwzględniane w długości podanej w polu *StrucLength*.

Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *ObjectRecOffset* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

PutMsgRecFields (MQLONG)

Można określić co najmniej jedną z następujących opcji:

MQPMRF_MSG_ID,

Pole identyfikatora komunikatu jest obecne.

MQPMRF_CORREL_ID

Pole identyfikatora korelacji jest obecne.

Identyfikator MQPMRF_GROUP_ID

Pole identyfikatora grupy jest obecne.

MQPMRF_FEEDBACK

Pole informacji zwrotnej jest obecne.

MQPMRF_ACCOUNTING_TOKEN,

Pole tokenu rozliczania jest obecne.

Jeśli nie ma żadnych pól MQPMR, podaj następujące informacje:

MQPMRF_NONE

Nie istnieją pola rekordu komunikatu umieszczonego w komunikacie. Parametr MQPMRF_NONE jest zdefiniowany w celu uzyskania dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest MQPMRF_NONE.

PutMsgRecOffset (MQLONG)

Daje to przesunięcie w bajtach pierwszego rekordu w tablicy rekordów komunikatów MQPMR, które zawierają właściwości komunikatu. Jeśli ta tablica jest obecna, w tej tablicy znajdują się rekordy *RecsPresent*. Rekordy te (plus wszystkie bajty pominięte między pierwszym rekordem umieszczania komunikatu a poprzednim polem) są uwzględniane w długości podanej w polu *StrucLength*.

Rekordy umieszczania komunikatów są opcjonalne; jeśli nie są dostępne żadne rekordy, wartość *PutMsgRecOffset* wynosi zero, a *PutMsgRecFields* ma wartość MQPMRF_NONE.

Wartością początkową tego pola jest 0.

RecsPresent (MQLONG)

To jest liczba miejsc docelowych. Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *RecsPresent* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

StrucId (MQCHAR4)

Wartość musi być następująca:

MQDH_STRUC_ID,

Identyfikator struktury nagłówka dystrybucji.

Dla języka programowania C jest również zdefiniowana stała zmienna MQDH_STRUC_ID_ARRAY; ma taką samą wartość jak MQDH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQDH_STRUC_ID.

StrucLength (MQLONG)

Jest to liczba bajtów od początku struktury MQDH do początku danych komunikatu zgodnie z tablicami rekordów MQOR i MQPMR. Dane są wykonywane w następującej kolejności:

- Struktura MQDH

- Tablica rekordów MQOR
- Tablica rekordów MQPMR
- Dane komunikatu

Tablice rekordów MQOR i MQPMR są adresowane przez przesunięcia zawarte w strukturze MQDH. Jeśli te przesunięcia powodują nieużywane bajty między jedną lub większą liczbą struktury MQDH, tablicami rekordów i danymi komunikatu, te nieużywane bajty muszą być uwzględnione w wartości *StrucLength*, ale treść tych bajtów nie jest zachowywana przez menedżer kolejek. Jest ona poprawna dla tablicy rekordów MQPMR w celu poprzedzania tablicy rekordów MQOR.

Wartością początkową tego pola jest 0.

Wersja (MQLONG)

Wartość musi być następująca:

MQDH_VERSION_1

Numer wersji struktury nagłówka dystrybucji.

Następująca stała określa numer wersji bieżącej wersji:

MQDH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka dystrybucji.

Początkowa wartość tego pola to MQDH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQDH

Tabela 34. Początkowe wartości pól w MQDH dla MQDH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQDH_STRUC_ID,	'DH↵↵'
<i>Version</i>	MQDH_VERSION_1	1
<i>StrucLength</i>	Brak	0
<i>Encoding</i>	Brak	0
<i>CodedCharSetId</i>	MQCCSI_UNDEFINED	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Flags</i>	MQDHF_NONE	0
<i>PutMsgRecFields</i>	MQPMRF_NONE	0
<i>RecsPresent</i>	Brak	0
<i>ObjectRecOffset</i>	Brak	0
<i>PutMsgRecOffset</i>	Brak	0
Uwagi: 1. Symbol ↵ reprezentuje pojedynczy pusty znak. 2. W języku programowania C: zmienna makraMQDH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQDH MyDH = {MQDH_DEFAULT};</pre>		

Deklaracja C

```

typedef struct tagMQDH MQDH;
struct tagMQDH {
    MQCHAR4  StrucId;          /* Structure identifier */
    MQLONG   Version;         /* Structure version number */
    MQLONG   StrucLength;     /* Length of MQDH structure plus following
                               MQOR and MQPMR records */
    MQLONG   Encoding;        /* Numeric encoding of data that follows
                               the MQOR and MQPMR records */
    MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows the MQOR and MQPMR records */
    MQCHAR8  Format;          /* Format name of data that follows the
                               MQOR and MQPMR records */
    MQLONG   Flags;           /* General flags */
    MQLONG   PutMsgRecFields; /* Flags indicating which MQPMR fields are
                               present */
    MQLONG   RecsPresent;     /* Number of MQOR records present */
    MQLONG   ObjectRecOffset; /* Offset of first MQOR record from start
                               of MQDH */
    MQLONG   PutMsgRecOffset; /* Offset of first MQPMR record from start
                               of MQDH */
};

```

Deklaracja języka COBOL

```

**      MQDH structure
10 MQDH.
**      Structure identifier
15 MQDH-STRUCID          PIC X(4).
**      Structure version number
15 MQDH-VERSION          PIC S9(9) BINARY.
**      Length of MQDH structure plus following MQOR and MQPMR records
15 MQDH-STRUCLength      PIC S9(9) BINARY.
**      Numeric encoding of data that follows the MQOR and MQPMR records
15 MQDH-ENCODING          PIC S9(9) BINARY.
**      Character set identifier of data that follows the MQOR and MQPMR
**      records
15 MQDH-CODEDCHARSETID   PIC S9(9) BINARY.
**      Format name of data that follows the MQOR and MQPMR records
15 MQDH-FORMAT           PIC X(8).
**      General flags
15 MQDH-FLAGS            PIC S9(9) BINARY.
**      Flags indicating which MQPMR fields are present
15 MQDH-PUTMSGRECFIELDS  PIC S9(9) BINARY.
**      Number of MQOR records present
15 MQDH-RECSPRESENT      PIC S9(9) BINARY.
**      Offset of first MQOR record from start of MQDH
15 MQDH-OBJECTRECOFFSET  PIC S9(9) BINARY.
**      Offset of first MQPMR record from start of MQDH
15 MQDH-PUTMSGRECOFFSET  PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQDH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Length of MQDH structure plus
                                   following MQOR and MQPMR
                                   records */
3 Encoding          fixed bin(31), /* Numeric encoding of data that
                                   follows the MQOR and MQPMR
                                   records */
3 CodedCharSetId    fixed bin(31), /* Character set identifier of data
                                   that follows the MQOR and MQPMR
                                   records */
3 Format            char(8),          /* Format name of data that follows
                                   the MQOR and MQPMR records */
3 Flags            fixed bin(31), /* General flags */
3 PutMsgRecFields   fixed bin(31), /* Flags indicating which MQPMR
                                   fields are present */
3 RecsPresent       fixed bin(31), /* Number of MQOR records present */
3 ObjectRecOffset   fixed bin(31), /* Offset of first MQOR record from
                                   start of MQDH */

```



```
3 PutMsgRecOffset fixed bin(31); /* Offset of first MQPMR record from
start of MQDH */
```

Wizualna deklaracja podstawowa

```
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-nagłówek Dead-letter

W poniższej tabeli podsumowano pola w strukturze.

Tabela 35. Pola w MQDLH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Reason</i>	Komunikat przyczyny dotarł do kolejki niedostarczonych komunikatów	Powód
<i>DestQName</i>	Nazwa oryginalnej kolejki docelowej	DestQName
<i>DestQMgrName</i>	Nazwa oryginalnego docelowego menedżera kolejek	Docelowy_menedżer_kolejek
<i>Encoding</i>	Kodowanie numeryczne danych, które są zgodne z MQDLH	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków dla danych, które są następujące: MQDLH	CodedCharSetId
<i>Format</i>	Nazwa formatu danych, które są następujące: MQDLH	Formatowanie
<i>PutApplType</i>	Typ aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów	Typ_aplikacji_wstawiającej
<i>PutApplName</i>	Nazwa aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów	Nazwa_aplikacji_wstawiającej
<i>PutDate</i>	Data umieszczenia komunikatu w kolejce niedostarczonych komunikatów	PutDate
<i>PutTime</i>	Czas umieszczenia komunikatu w kolejce niedostarczonych komunikatów	PutTime

Przegląd produktu MQDLH

Dostępność: Wszystkie platformy IBM MQ .

Cel: Struktura MQDLH opisuje informacje, które prefikują dane komunikatu aplikacji dla komunikatów znajdujących się w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Komunikat może pojawić się w kolejce niedostarczonych komunikatów, ponieważ menedżer kolejek lub agent kanału komunikatów przekierował go do kolejki lub ponieważ aplikacja umieściła komunikat bezpośrednio w kolejce.

Nazwa formatu: MQFMT_DEAD_LETTER_HEADER.

Zestaw znaków i kodowanie: Pola w strukturze MQDLH znajdują się w zestawie znaków i kodowaniu podanym w polach *CodedCharSetId* i *Encoding* . Są one określane w strukturze nagłówka poprzedzającym wartość MQDLH lub w strukturze MQMD, jeśli wartość MQDLH jest na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Jeśli używane są klasy WMQ dla Java/JMS, a strona kodowa zdefiniowana w strukturze MQMD nie jest obsługiwana przez maszynę wirtualną Java , to wartość MQDLH jest zapisywana w zestawie znaków UTF-8 .

Użycie: Aplikacje, które umieszczają komunikaty bezpośrednio w kolejce niedostarczonych komunikatów, muszą prefiksować dane komunikatu ze strukturą MQDLH i inicjować pola odpowiednimi wartościami. Jednak menedżer kolejek nie wymaga, aby struktura MQDLH była obecna lub że określono poprawne wartości dla pól.

Jeśli komunikat jest zbyt długi, aby można go było umieścić w kolejce niedostarczonych komunikatów, aplikacja musi wykonać jedną z następujących czynności:

- Obetnij dane komunikatu, aby zmieściły się w kolejce niedostarczonych komunikatów.
- Zapisz komunikat w pamięci dyskowej i umieść komunikat raportu o wyjątku w kolejce niedostarczonych komunikatów, co wskazuje na to.
- Usuń komunikat i zwróć błąd do jego inicjatora. Jeśli komunikat jest (lub może być) komunikatem krytycznym, należy wykonać tę wiadomość tylko wtedy, gdy wiadomo, że inicjator nadal ma kopię komunikatu, na przykład komunikat odebrany przez agenta kanału komunikatów z kanału komunikacyjnego.

Które z powyższych działań jest odpowiednie (jeśli takie istnieją) zależy od projektu aplikacji.

Menedżer kolejek wykonuje specjalne przetwarzanie, gdy komunikat, który jest segmentem, jest umieszczany w strukturze MQDLH z przodu. Szczegółowe informacje można znaleźć w opisie struktury MQMDE.

umieszczanie komunikatów w kolejce niedostarczonych komunikatów: Gdy komunikat jest umieszczany w kolejce niedostarczonych komunikatów, struktura MQMD używana dla wywołania MQPUT lub MQPUT1 musi być identyczna z wartością MQMD powiązaną z komunikatem (zwykle jest to MQMD zwrócony przez wywołanie MQGET), z wyjątkiem następujących:

- Ustaw pola *CodedCharSetId* i *Encoding* na dowolny zestaw znaków i kodowanie, które są używane dla pól w strukturze MQDLH.
- Ustaw pole *Format* na wartość MQFMT_DEAD_LETTER_HEADER, aby wskazać, że dane rozpoczynają się od struktury MQDLH.
- Ustaw pola kontekstu (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime*, *UserIdentifier*), korzystając z opcji kontekstu odpowiedniej dla okoliczności:
 - Aplikacja umieszczająca w kolejce niedostarczonych komunikatów komunikat, który nie jest powiązany z żadnym poprzedzającym komunikatem, musi używać opcji MQPMO_DEFAULT_CONTEXT. Powoduje to, że menedżer kolejek ustawi wszystkie pola kontekstu w deskrypcorze komunikatu na wartości domyślne.

- Aplikacja serwera umieszczając w kolejce niedostarczonych komunikatów komunikat, który został właśnie odebrany, musi użyć opcji MQPMO_PASS_ALL_CONTEXT, aby zachować oryginalne informacje o kontekście.
- Aplikacja serwera umieszczając w kolejce niedostarczonych komunikatów *odpowiedź* na odebraną wiadomość musi używać opcji MQPMO_PASS_IDENTITY_CONTEXT. Ta opcja zachowuje informacje o tożsamości, ale ustawia informacje o pochodzeniu, które mają być informacjami o aplikacji serwera.
- Agent kanału komunikatów umieszczający w kolejce niedostarczonych komunikatów komunikat, który został odebrany z kanału komunikacyjnego, musi użyć opcji MQPMO_SET_ALL_CONTEXT, aby zachować oryginalne informacje o kontekście.

W samej strukturze MQDLH ustaw pola w następujący sposób:

- Ustaw pola *CodedCharSetId*, *Encodingi Format* na wartości opisujące dane, które są zgodne ze strukturą MQDLH, zwykle wartości z oryginalnego deskryptora komunikatu.
- Ustaw pola kontekstu *PutApplType*, *PutApplName*, *PutDatei PutTime* na wartości odpowiednie dla aplikacji, która umieszcza komunikat w kolejce niedostarczonych komunikatów. Te wartości nie są powiązane z pierwotnym komunikatem.
- W razie potrzeby ustaw inne pola.

Upewnij się, że wszystkie pola mają poprawne wartości, a pola znakowe są dopełniane spacjami do zdefiniowanej długości pola. Nie należy kończyć przedwcześnie danych znakowych przy użyciu znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca wartości NULL i kolejnych znaków w puste znaki w strukturze MQDLH.

Pobieranie komunikatów z kolejki niedostarczonych komunikatów: Aplikacje, które pobierają komunikaty z kolejki niedostarczonych komunikatów, muszą sprawdzić, czy komunikaty zaczynają się od struktury MQDLH. Aplikacja może określić, czy struktura MQDLH jest obecna, badając pole *Format* w deskrypcorze komunikatu MQMD. Jeśli pole ma wartość MQFMT_DEAD_LETTER_HEADER, dane komunikatu zaczynają się od struktury MQDLH. Należy pamiętać, że komunikaty, które aplikacje pochodzą z kolejki niedostarczonych komunikatów, mogą zostać obcięte, jeśli były one pierwotnie zbyt długie dla kolejki.

Pola dla MQDLH

Struktura MQDLH zawiera następujące pola: pola są opisane w **kolejności alfabetycznej** :

CodedCharSetId (MQLONG)

CodedCharSetId jest identyfikatorem zestawu znaków danych, który przepływa przez strukturę MQDLH (zwykle są to dane z oryginalnego komunikatu); nie ma zastosowania do danych znakowych w samej strukturze MQDLH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych po tej strukturze znajdują się w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie można użyć komendy MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcorze MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

DestQMGrNazwa (MQCHAR48)

DestQMGrNazwa to nazwa menedżera kolejek, który był oryginalnym miejscem docelowym dla komunikatu.

Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

DestQName (MQCHAR48)

DestQName to nazwa kolejki komunikatów, która była oryginalnym miejscem docelowym dla komunikatu.

Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Kodowanie (MQLONG)

Kodowanie jest kodowaniem liczbowym danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu); nie ma ono zastosowania do danych liczbowych w samej strukturze MQDLH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

Format (MQCHAR8)

Format jest nazwą formatu danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak reguły kodowania pola *Format* w strukturze MQMD.

Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Nazwa PutAppl(MQCHAR28)

PutApplNazwa jest nazwą aplikacji, która umiała komunikat w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format nazwy zależy od pola *PutApplType*. Format może się różnić w zależności od wersji. Zapoznaj się z opisem pola *PutApplName* w [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Jeśli menedżer kolejek przekierowuje komunikat do kolejki niedostarczonych komunikatów, program *PutApplName* zawiera pierwsze 28 znaków nazwy menedżera kolejek. Jeśli to konieczne, uzupełnione odstępami.

Długość tego pola jest podana przez wartość MQ_PUT_APPL_NAME_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C i 28 pustych znaków w innych językach programowania.

Typ PutAppl(MQLONG)

PutApplTyp to typ aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

To pole ma takie samo znaczenie, jak pole *PutApplType* w deskrytorze komunikatu MQMD (szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 417).

Jeśli menedżer kolejek przekierowuje komunikat do kolejki niedostarczonych komunikatów, parametr *PutApplType* ma wartość MQAT_QMGR.

Wartością początkową tego pola jest 0.

PutDate (MQCHAR8)

PutDate to data umieszczenia komunikatu w kolejce niedostarczonych komunikatów (undelivered-message).

Format używany dla daty, w której to pole jest generowane przez menedżer kolejek:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr

rok (cztery cyfry)

MM

miesiąc w roku (01 do 12)

DD

Dzień miesiąca (01 do 31)

Czas Greenwich (GMT) jest używany dla pól *PutDate* i *PutTime*, pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Długość tego pola jest podana przez wartość MQ_PUT_DATE_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C i osiem znaków odstępu w innych językach programowania.

PutTime (MQCHAR8)

PutTime to czas, w którym komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format używany w czasie, gdy pole jest generowane przez menedżer kolejek:

- GGMMSSSTH

gdzie znaki reprezentują:

GG

godzin (od 00 do 23)

MM

minuty (od 00 do 59)

SS

sekundy (od 00 do 59; patrz uwaga)

T

Dziesiątych sekundy (od 0 do 9)

H

Setnych części sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, można w rzadkich przypadkach zwrócić 60 lub 61 sekund w ciągu sekundy w składce *PutTime*. Dzieje się tak wtedy, gdy sekundy przestępne są wstawiane w globalnym standardzie czasu.

Czas Greenwich (GMT) jest używany dla pól *PutDate* i *PutTime*, pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Długość tego pola jest podana przez wartość MQ_PUT_TIME_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C i osiem znaków odstępu w innych językach programowania.

Przyczyna (MQLONG)

Pole Przyczyna określa przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów, a nie w oryginalnej kolejce docelowej.

Określa przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów, a nie w oryginalnej kolejce docelowej. Powinien to być jeden z wartości MQFB_* lub MQRC_* (na przykład MQRC_Q_FULL). Aby uzyskać szczegółowe informacje na temat wspólnych wartości MQFB_*, które mogą wystąpić, należy zapoznać się z opisem w polu *Feedback* w publikacji [“MQMD-deskryptor komunikatu” na stronie 417](#).

Jeśli wartość znajduje się w zakresie MQFB_IMS_FIRST za pomocą MQFB_IMS_LAST, rzeczywisty kod błędu IMS może zostać określony przez odjęcie wartości MQFB_IMS_ERROR od wartości pola *Reason*.

Niektóre wartości MQFB_* występują tylko w tym polu. Odnoszą się one do komunikatów repozytorium, komunikatów wyzwalacza lub komunikatów kolejki transmisji, które zostały przesłane do kolejki niedostarczonych komunikatów. Są to:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')

Aplikacja przetwarzająca komunikat wyzwalacza nie może uruchomić aplikacji o nazwie określonej w polu *ApplId* komunikatu wyzwalacza (patrz [“MQTM-komunikat wyzwalacza” na stronie 603](#)).

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza.

MQFB_APPL_TYPE_ERROR (X'0000010B')

Aplikacja przetwarzająca komunikat wyzwalacza nie może uruchomić aplikacji, ponieważ pole *ApplType* komunikatu wyzwalacza nie jest poprawne (patrz [“MQTM-komunikat wyzwalacza” na stronie 603](#)).

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')

Komunikat został wyświetlony w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE przeznaczona dla kolejki klastra, która została otwarta za pomocą opcji MQOO_BIND_ON_OPEN, ale zdalny kanał odbierający klastry, który ma być używany do przesyłania komunikatu do kolejki docelowej, został usunięty przed wysłaniem komunikatu. Ponieważ określono parametr MQOO_BIND_ON_OPEN, do przesyłania komunikatu można użyć tylko kanału wybranego, gdy kolejka została otwarta. Ponieważ ten kanał nie jest już dostępny, komunikat jest umieszczany w kolejce niedostarczonych komunikatów.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')

Komunikat nie jest komunikatem repozytorium.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')

Komunikat został zatrzymany przez wyjście automatycznej definicji kanału.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')

Komunikat został zatrzymany przez wyjście komunikatów kanału.

MQFB_TM_ERROR (X'0000010A')

Pole *Format* w strukturze MQMD określa wartość MQFMT_TRIGGER, ale komunikat nie rozpoczyna się od poprawnej struktury MQTM. Na przykład *StrucId* mnemonik eye-catcher może nie być poprawny, *Version* może nie zostać rozpoznany, albo długość komunikatu wyzwalacza może być niewystarczająca do zmniejszenia struktury MQTM.

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza i może wygenerować ten kod informacji zwrotnych.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')

Agent kanału komunikatów stwierdził, że komunikat w kolejce transmisji nie jest w poprawnym formacie. Agent kanału komunikatów umieszcza komunikat w kolejce niedostarczonych komunikatów przy użyciu tego kodu informacji zwrotnych.

Jedną z częstych przyczyn jest to, że komunikat został umieszczony bezpośrednio w kolejce transmisji, dlatego komunikat nie ma oczekiwanego nagłówka XQH. Komunikaty powinny być umieszczane w kolejce transmisji przy użyciu kolejki zdalnej, chyba że aplikacja buduje nagłówki MQXQH.

Wartością początkową tego pola jest MQRC_NONE.

StrucId (MQCHAR4)

StrucId to identyfikator struktury.

Wartość musi być następująca:

MQDLH_STRUC_ID

Identyfikator struktury nagłówka niedostarczonych komunikatów.

Dla języka programowania C jest również zdefiniowana stała MQDLH_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQDLH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQDLH_STRUC_ID.

Wersja (MQLONG)

Wersja jest numerem wersji struktury.

Wartość musi być następująca:

MQDLH_VERSION_1

Numer wersji dla struktury nagłówka niedostarczonych komunikatów.

Następująca stała określa numer wersji bieżącej wersji:

MQDLH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka niedostarczonych komunikatów.

Początkowa wartość tego pola to MQDLH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQDLH

Tabela 36. Początkowe wartości pól w MQDLH dla MQDLH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQDLH_STRUC_ID	'DLH↵'
<i>Version</i>	MQDLH_VERSION_1	1
<i>Reason</i>	MQRC_NONE	0
<i>DestQName</i>	Brak	Pusty łańcuch lub odstęp
<i>DestQMgrName</i>	Brak	Pusty łańcuch lub odstęp
<i>Encoding</i>	Brak	0
<i>CodedCharSetId</i>	MQCCSI_UNDEFINED	0
<i>Format</i>	MQFMT_NONE	Puste
<i>PutApplType</i>	Brak	0
<i>PutApplName</i>	Brak	Pusty łańcuch lub odstęp
<i>PutDate</i>	Brak	Pusty łańcuch lub odstęp
<i>PutTime</i>	Brak	Pusty łańcuch lub odstęp
Uwagi: - Symbol ↵ reprezentuje pojedynczy pusty znak. - Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. - W języku programowania C: zmienna makraParametr MQDLH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQDLH MyDLH = {MQDLH_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
```

```

MQCHAR4   StrucId;           /* Structure identifier */
MQLONG    Version;          /* Structure version number */
MQLONG    Reason;           /* Reason message arrived on dead-letter
                             (undelivered-message) queue */
MQCHAR48   DestQName;        /* Name of original destination queue */
MQCHAR48   DestQMgrName;     /* Name of original destination queue
                             manager */
MQLONG     Encoding;         /* Numeric encoding of data that follows
                             MQDLH */
MQLONG     CodedCharSetId;   /* Character set identifier of data that
                             follows MQDLH */
MQCHAR8    Format;           /* Format name of data that follows
                             MQDLH */
MQLONG     PutApplType;      /* Type of application that put message on
                             dead-letter (undelivered-message)
                             queue */
MQCHAR28   PutApplName;      /* Name of application that put message on
                             dead-letter (undelivered-message)
                             queue */
MQCHAR8     PutDate;         /* Date when message was put on dead-letter
                             (undelivered-message) queue */
MQCHAR8     PutTime;         /* Time when message was put on the
                             dead-letter (undelivered-message)
                             queue */
};

```

Deklaracja języka COBOL

```

**  MQDLH structure
10 MQDLH.
**  Structure identifier
15 MQDLH-STRUCID      PIC X(4).
**  Structure version number
15 MQDLH-VERSION     PIC S9(9) BINARY.
**  Reason message arrived on dead-letter (undelivered-message) queue
15 MQDLH-REASON      PIC S9(9) BINARY.
**  Name of original destination queue
15 MQDLH-DESTQNAME   PIC X(48).
**  Name of original destination queue manager
15 MQDLH-DESTQMGRNAME PIC X(48).
**  Numeric encoding of data that follows MQDLH
15 MQDLH-ENCODING    PIC S9(9) BINARY.
**  Character set identifier of data that follows MQDLH
15 MQDLH-CODEDCHARSETID PIC S9(9) BINARY.
**  Format name of data that follows MQDLH
15 MQDLH-FORMAT      PIC X(8).
**  Type of application that put message on dead-letter
**  (undelivered-message) queue
15 MQDLH-PUTAPPLTYPE PIC S9(9) BINARY.
**  Name of application that put message on dead-letter
**  (undelivered-message) queue
15 MQDLH-PUTAPPLNAME PIC X(28).
**  Date when message was put on dead-letter (undelivered-message)
**  queue
15 MQDLH-PUTDATE     PIC X(8).
**  Time when message was put on the dead-letter (undelivered-message)
**  queue
15 MQDLH-PUTTIME     PIC X(8).

```

Deklaracja PL/I

```

dcl
1 MQDLH based,
3 StrucId      char(4),           /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Reason       fixed bin(31),    /* Reason message arrived on
                                dead-letter (undelivered-message)
                                queue */
3 DestQName     char(48),         /* Name of original destination
                                queue */
3 DestQMgrName  char(48),         /* Name of original destination queue
                                manager */
3 Encoding      fixed bin(31),    /* Numeric encoding of data that
                                follows MQDLH */
3 CodedCharSetId fixed bin(31),  /* Character set identifier of data
                                that follows MQDLH */

```



```

3 Format      char(8),      /* Format name of data that follows
                             MQDLH */
3 PutApplType fixed bin(31), /* Type of application that put
                             message on dead-letter
                             (undelivered-message) queue */
3 PutApplName char(28),     /* Name of application that put
                             message on dead-letter
                             (undelivered-message) queue */
3 PutDate     char(8),      /* Date when message was put on
                             dead-letter (undelivered-message)
                             queue */
3 PutTime     char(8);      /* Time when message was put on the
                             dead-letter (undelivered-message)
                             queue */

```

Deklaracja High Level Assembler

```

MQDLH          DSECT
MQDLH_STRUCID  DS      CL4   Structure identifier
MQDLH_VERSION  DS      F     Structure version number
MQDLH_REASON   DS      F     Reason message arrived on dead-letter
*              (undelivered-message) queue
MQDLH_DESTQNAME DS     CL48   Name of original destination queue
MQDLH_DESTQMGNAME DS     CL48 Name of original destination queue
*              manager
MQDLH_ENCODING DS      F     Numeric encoding of data that follows
*              MQDLH
MQDLH_CODEDCCHARSETID 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)

```

Wizualna deklaracja podstawowa

```

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-usuwanie opcji uchwytu komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 37. Pola w MQDMHO		
Pole	Opis	Temat
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
Options	Opcje	Opcje

Przegląd produktu MQDMHO

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ .

Cel: Struktura MQDMHO umożliwia aplikacjom określanie opcji, które sterują sposobem usuwania uchwytów komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQDLTMH .

Zestaw znaków i kodowanie: Dane w programie MQDMHO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla MQDMHO

Struktura MQDMHO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

Wartość musi być następująca:

MQDMHO_NONE

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQDMHO_NONE.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQDMHO_STRUC_ID

Identyfikator struktury opcji uchwytu komunikatu usuwania.

Dla języka programowania w języku C jest również zdefiniowana stała MQDMHO_STRUC_ID_ARRAY .
Ma ona taką samą wartość jak MQDMHO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQDMHO_STRUC_ID.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQDMHO_VERSION_1

Version-1 -usuń strukturę opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQDMHO_CURRENT_VERSION

Bieżąca wersja struktury opcji uchwytu komunikatu usuwania.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQDMHO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQDMHO

Tabela 38. Początkowe wartości pól w MQDMHO

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQDMHO_STRUC_ID	' DMHO '
Version	MQDMHO_VERSION_1	1
Options	MQDMHO_NONE	0

Uwagi:

1. W języku programowania C: zmienna makraParametr MQDMHO_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQDLTMH */
};
```

Deklaracja języka COBOL

```
**      MQDMHO structure
10 MQDMHO.
**      Structure identifier
15 MQDMHO-STRUCID      PIC X(4).
**      Structure version number
15 MQDMHO-VERSION      PIC S9(9) BINARY.
**      Options that control the action of MQDLTMH
15 MQDMHO-OPTIONS      PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQDMHO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Options      fixed bin(31),    /* Options that control the action of MQDLTMH */
```

Deklaracja High Level Assembler

```
MQDMHO          DSECT
MQDMHO_STRUCID  DS    CL4      Structure identifier
MQDMHO_VERSION  DS    F        Structure version number
MQDMHO_OPTIONS  DS    F        Options that control the action of
*                                MQDLTMH
MQDMHO_LENGTH   EQU    *-MQDMHO
MQDMHO_AREA     DS    CL(MQDMHO_LENGTH)
```

MQDMPO-opcje usuwania właściwości komunikatu

W poniższej tabeli podsumowano pola w strukturze. Struktura MQDMPO-opcje właściwości usuwania komunikatu

Tabela 39. Pola w MQDMPO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem MQDMPO	Opcje

Przegląd produktu MQDMPO

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ .

Przeznaczenie: Struktura MQDMPO umożliwia aplikacjom określanie opcji sterujących sposobem usuwania właściwości komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQDLTMP.

Zestaw znaków i kodowanie: Dane w tabeli MQDMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla MQDMPO

Usuń strukturę opcji właściwości komunikatu-pola

Struktura MQDMPO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

Usuń strukturę opcji właściwości komunikatu-pole Opcje

Opcje lokalizacji: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości.

MQDMPO_DEL_FIRST

Usuwa pierwszą właściwość, która jest zgodna z podaną nazwą.

MQDMPO_DEL_PROP_UNDER_CURSOR

Usuwa właściwość wskazywaną przez kursor właściwości. Jest to właściwość, która została ostatnio sprawdzona przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości zostanie zresetowany, gdy uchwyt komunikatu zostanie ponownie wykorzystany. Jest ona również resetowana, gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony, wywołanie zakończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyna MQRC_PROPERTY_NOT_AVAILABLE. Jeśli właściwość wskazywana przez kursor właściwości została już usunięta, wywołanie nie powiedzie się również z kodem zakończenia MQCC_FAILED i przyczyna MQRC_PROPERTY_NOT_AVAILABLE.

Jeśli żaden z opcji thees nie jest wymagany, można użyć następującej opcji:

MQDMPO_NONE

Nie określono żadnych opcji.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest MQDMPO_DEL_FIRST.

StrucId (MQCHAR4)

Usuń strukturę opcji właściwości komunikatu-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQDMPO_STRUC_ID,

Identyfikator struktury opcji usuwania właściwości komunikatu.

Dla języka programowania C jest również zdefiniowana stała MQDMPO_STRUC_ID_ARRAY. Ma ona taką samą wartość co identyfikator MQDMPO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQDMPO_STRUC_ID.

Wersja (MQLONG)

Usuń strukturę opcji właściwości komunikatu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQDMPO_VERSION_1

Numer wersji dla struktury opcji usuwania właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQDMPO_CURRENT_VERSION

Bieżąca wersja struktury opcji usuwania właściwości komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQDMPO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQDMPO

Struktura opcji usuwania właściwości komunikatu-wartości początkowe

Tabela 40. Początkowe wartości pól w MQDMPO		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQDMPO_STRUC_ID,	'DMPO '
Version	MQDMPO_VERSION_1	1
Options	Opcje sterujące działaniem komendy MQDLTMP	MQDMPO_NONE
Uwagi: 1. W języku programowania C: zmienna makraParametr MQDMPO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQDMPO MyDMPO = {MQDMPO_DEFAULT};</pre>		

Deklaracja C

Usuń strukturę opcji właściwości komunikatu-deklaracja języka C

```
typedef struct tagMQDMPO MQDMPO;  
struct tagMQDMPO {  
    MQLONG StrucId;           /* Structure identifier */  
    MQLONG Version;          /* Structure version number */  
    MQLONG Options;          /* Options that control the action of  
                             MQDLTMP */  
};
```

Deklaracja języka COBOL

Usuń strukturę opcji właściwości komunikatu-deklaracja języka COBOL

```
** MQDMPO structure  
10 MQDMPO.  
** Structure identifier  
15 MQDMPO-STRUCID          PIC X(4).  
** Structure version number  
15 MQDMPO-VERSION          PIC S9(9) BINARY.  
** Options that control the action of MQDLTMP  
15 MQDMPO-OPTIONS          PIC S9(9) BINARY.
```

Deklaracja PL/I

Usuń strukturę opcji właściwości komunikatu-deklaracja języka PL/I

```

Dcl
  1 MQDPMO based,
  3 StrucId      char(4),          /* Structure identifier */
  3 Version      fixed bin(31), /* Structure version number */
  3 Options      fixed bin(31), /* Options that control the action
                                of MQDLTMP */

```

Deklaracja High Level Assembler

Usuń strukturę opcji właściwości komunikatu-Składanie deklaracji języka

```

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-osadzony nagłówek PCF

W poniższej tabeli podsumowano pola w strukturze.

Tabela 41. Pola w tabeli MQEPH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQEPH oraz struktury MQCFH i struktur parametrów, które są zgodne z tym	StrucLength
<i>Encoding</i>	Kodowanie numeryczne danych, które są zgodne z ostatnią strukturą parametru PCF	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków danych, który jest zgodny z ostatnią strukturą parametru PCF	CodedCharSetId
<i>Format</i>	Format nazwy danych, które są następujące po ostatniej strukturze parametru PCF	Formatowanie
<i>Flags</i>	Flagi	Flagi
<i>PCFHeader</i>	Nagłówek PCF (Programmable command format)	Nagłówek PCFHeader

Przegląd produktu MQEPH

Dostępność: Wszystkie platformy IBM MQ .

Cel: Struktura MQEPH opisuje dodatkowe dane, które są obecne w komunikacie, gdy jest to komunikat PCF (programmable command format). Pole *PCFHeader* definiuje parametry PCF, które są zgodne z tą strukturą, co pozwala na śledzenie danych komunikatu PCF z innymi nagłówkami.

Nazwa formatu: MQFMT_EMBEDDED_PCF

Zestaw znaków i kodowanie: Dane w tabeli MQEPH muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie wartości MQEPH w polach *CodedCharSetId* i *Encoding* w:

- MQMD (jeśli struktura MQEPH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQEPH (wszystkie inne obserwacje).

Użycie: Nie można użyć struktur MQEPH do wysyłania komend do serwera komend lub innego serwera akceptowanego przez PCF menedżera kolejek.

Podobnie serwer komend lub inny serwer akceptujący menedżera kolejek PCF nie generują odpowiedzi ani zdarzeń zawierających struktury MQEPH.

Pola dla tabeli MQEPH

Struktura MQEPH zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**:

CodedCharSetId (MQLONG)

Jest to identyfikator zestawu znaków danych, który jest zgodny ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma on zastosowania do danych znakowych w samej strukturze MQEPH.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Kodowanie (MQLONG)

Jest to kodowanie liczbowe dla danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie dotyczy to danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest 0.

Flagi (MQLONG)

Dozwolone są następujące wartości:

MQEPH_NONE

Nie określono żadnych flag. Opcja MQEPH_NONE jest zdefiniowana w celu uzyskania dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

MQEPH_CCSID_EMBEDDED,

Zestaw znaków parametrów zawierających dane znakowe jest określany indywidualnie w obrębie pola CodedCharSetId w każdej strukturze. Zestaw znaków pól StrucId i Format jest zdefiniowany przez pole CodedCharSetId w strukturze nagłówka poprzedzającej strukturę MQEPH lub przez pole CodedCharSetId w strukturze MQMD, jeśli wartość MQEPH jest na początku komunikatu.

Wartością początkową tego pola jest MQEPH_NONE.

Format (MQCHAR8)

Jest to nazwa formatu danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF.

Wartością początkową tego pola jest MQFMT_NONE.

PCFHeader (MQCFH)

Jest to nagłówek programowalnego formatu komend (PCF), definiujący parametry PCF, które są zgodne ze strukturą MQEPH. Dzięki temu można śledzić dane komunikatu PCF z innymi nagłówkami.

Nagłówek PCF jest początkowo zdefiniowany z następującymi wartościami:

<i>Tabela 42. Początkowe wartości pól w MQCFH</i>		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>Type</i>	MQCFT_NONE	0
<i>StrucLength</i>	MQCFH_STRUC_LENGTH	36
<i>Version</i>	MQCFH_VERSION_3	3
<i>StrucLength</i>	Brak	0

Tabela 42. Początkowe wartości pól w MQCFH (kontynuacja)		
Nazwa pola	Nazwa stałej	Wartość stałej
Command	MQCMD_BRAK	0
MsgSeqNumber	Brak	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	Brak	0

Aplikacja musi zmienić wartość Type z MQCFT_NONE na poprawny typ struktury dla użycia, który jest używany w osadzonym nagłówku PCF.

StrucId (MQCHAR4)

Wartość musi być następująca:

IDENTYFIKATOR_STRUKTURY_MQL

Identyfikator struktury nagłówka dystrybucji.

Dla języka programowania C jest również zdefiniowana stała MQEPH_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQDH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wartością początkową tego pola jest MQEPH_STRUC_ID.

StrucLength (MQLONG)

Jest to ilość danych poprzedzająca następną strukturę nagłówka. Obejmuje ona:

- Długość nagłówka MQEPH
- Długość wszystkich parametrów PCF następujących po nagłówku
- Każde puste dopełnienie po tych parametrach

StrucLength musi być wielokrotnością 4.

Stała długość części struktury jest definiowana przez wartość MQEPH_STRUC_LENGTH_FIXED.

Wartością początkową tego pola jest 68.

Wersja (MQLONG)

Wartość musi być następująca:

MQEPH_VERSION_1

Numer wersji dla osadzonej struktury nagłówka PCF.

Następująca stała określa numer wersji bieżącej wersji:

MQCFH_VERSION_3

Bieżąca wersja wbudowanej struktury nagłówka PCF.

Początkowa wartość tego pola to MQEPH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQEPH

Tabela 43. Początkowe wartości pól w MQEPH dla MQEPH		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	IDENTYFIKATOR_STRUKTURY_MQL	'EPH'
Version	MQEPH_VERSION_1	1

Tabela 43. Początkowe wartości pól w MQEPH dla MQEPH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
StrucLength	MQEPH_STRUC_LENGTH_FIXED	68
Encoding	Brak	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Puste
Flags	MQEPH_NONE	0
PCFHeader	Nazwy i wartości zdefiniowane w produkcie Tabela 42 na stronie 363	0

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraParametr MQEPH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    StrucLength;      /* Total length of MQEPH including the MQCFH
                                and parameter structures that follow it */
    MQLONG    Encoding;        /* Numeric encoding of data that follows last
                                PCF parameter structure */
    MQLONG    CodedCharSetId;   /* Character set identifier of data that
                                follows last PCF parameter structure */
    MQCHAR8   Format;          /* Format name of data that follows last PCF
                                parameter structure */
    MQLONG    Flags;           /* Flags */
    MQCFH     PCFHeader;       /* Programmable command format header */
};
```

Deklaracja języka COBOL

```
**  MQEPH structure
10 MQEPH.
**    Structure identifier
15 MQEPH-STRUCID      PIC X(4).
**    Structure version number
15 MQEPH-VERSION      PIC S9(9) BINARY.
**    Total length of MQEPH structure including the MQCFH
**    and parameter structures that follow it
15 MQEPH-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-STRULENGTH    PIC S9(9) BINARY.
**      Structure version number
20 MQEPH-PCFHEADER-VERSION       PIC S9(9) BINARY.
**      Command identifier
20 MQEPH-PCFHEADER-COMMAND       PIC S9(9) BINARY.
**      Message sequence number
20 MQEPH-PCFHEADER-MSGSEQNUMBER  PIC S9(9) BINARY.
**      Control options
20 MQEPH-PCFHEADER-CONTROL       PIC S9(9) BINARY.
**      Completion code
20 MQEPH-PCFHEADER-COMPCODE      PIC S9(9) BINARY.
**      Reason code qualifying completion code
20 MQEPH-PCFHEADER-REASON        PIC S9(9) BINARY.
**      Count of parameter structures
20 MQEPH-PCFHEADER-PARAMETERCOUNT PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQEPH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31), /* Structure version number */
3 StrucLength      fixed bin(31), /* Total Length of MQEPH including the
                                   MQCFH and parameter structures that
                                   follow it
3 Encoding          fixed bin(31), /* Numeric encoding of data that follows
                                   last PCF parameter structure
3 CodedCharSetId   fixed bin(31), /* Character set identifier of data that
                                   follows last PCF parameter structure
3 Format            char(8),          /* Format name of data that follows last
                                   PCF parameter structure */
3 Flags            fixed bin(31), /* Flags */
3 PCFHeader,       /* Programmable command format header
5 Type             fixed bin(31), /* Structure type */
5 StrucLength      fixed bin(31), /* Structure length */
5 Version          fixed bin(31), /* Structure version number */
5 Command          fixed bin(31), /* Command identifier */
5 MsgseqNumber     fixed bin(31), /* Message sequence number */
5 Control          fixed bin(31), /* Control options */
5 CompCode         fixed bin(31), /* Completion code */
5 Reason           fixed bin(31), /* Reason code qualifying completion code */
5 ParameterCount   fixed bin(31); /* Count of parameter structures */

```

Deklaracja High Level Assembler

MQEPH	DSECT	
MQEPH_STRUCID	DS	CL4 Structure identifier
MQEPH_VERSION	DS	F Structure version number
MQEPH_STRULENGTH	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_STRULENGTH	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

MQEPH_AREA

 ORG MQEPH
 DS CL (MQEPH_LENGTH)

Wizualna deklaracja podstawowa

```

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-Opcje Get-message

W poniższej tabeli podsumowano pola w strukturze.

Tabela 44. Pola w MQGMO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem MQGET	MQGMO-pole opcji
<i>WaitInterval</i>	Interwał oczekiwania	WaitInterval
<i>Signal1</i>	Przerwany przez sygnał ze skryptu	Signal1
<i>Signal2</i>	Identyfikator sygnału	Signal2
<i>ResolvedQName</i>	Rozstrzygnięta nazwa kolejki docelowej	ResolvedQName
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQGMO_VERSION_2.		
<i>MatchOptions</i>	Opcje kontrolujące kryteria wyboru używane dla operacji MQGET	MatchOptions
<i>GroupStatus</i>	Flaga wskazująca, czy wczytany komunikat znajduje się w grupie	GroupStatus
<i>SegmentStatus</i>	Flaga wskazująca, czy pobrano komunikat, czy jest to segment komunikatu logicznego	SegmentStatus
<i>Segmentation</i>	Flaga wskazująca, czy dalsza segmentacja jest dozwolona dla pobranego komunikatu	Segmentacja
<i>Reserved1</i>	Zarezerwowane	Reserved1
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQGMO_VERSION_3.		
<i>MsgToken</i>	Token komunikatu	MsgToken
<i>ReturnedLength</i>	Długość zwracanych danych komunikatu (w bajtach)	ReturnedLength

Tabela 44. Pola w MQGMO (kontynuacja)		
Pole	Opis	Temat
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQGMO_VERSION_4.		
<i>Reserved2</i>	Zarezerwowane	<u>Reserved2</u>
<i>MsgHandle</i>	Uchwyt do komunikatu, który ma zostać wypełniony właściwościami komunikatu pobieranego z kolejki.	<u>MsgHandle</u>

Przegląd produktu MQGMO

Dostępność: Wszystkie platformy IBM MQ .

Cel: Struktura MQGMO umożliwia aplikacji sterowanie sposobem usuwania komunikatów z kolejek. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQGET.

Wersja: Bieżąca wersja produktu MQGMO to MQGMO_VERSION_4. Niektóre pola są dostępne tylko w niektórych wersjach produktu MQGMO. Jeśli zachodzi potrzeba obsługi portów aplikacji między kilkoma środowiskami, należy upewnić się, że wersja produktu MQGMO jest spójna we wszystkich środowiskach. Pola, które istnieją tylko w określonych wersjach struktury, są identyfikowane jako takie w [“MQGMO-Opcje Get-message”](#) na stronie 367 i w opisach pól.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQGMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQGMO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , należy ustawić pole *Version* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie: Dane w produkcie MQGMO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla produktu MQGMO

Struktura MQGMO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

GroupStatus (MQCHAR)

Ta opcja wskazuje, czy pobrany komunikat znajduje się w grupie.

Ma jedną z następujących wartości:

MQGS_NOT_IN_GROUP

Komunikat nie znajduje się w grupie.

MQGS_MSG_IN_GROUP

Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

MQGS_LAST_MSG_IN_GROUP

Komunikat jest ostatnim w grupie.

Jest to również wartość zwracana, jeśli grupa składa się tylko z jednego komunikatu.

To jest pole wyjściowe. Początkowa wartość tego pola to MQGS_NOT_IN_GROUP. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

MatchOptions (MQLONG)

Te opcje umożliwiają aplikacji wybór pól w parametrze *MsgDesc* , które mają być używane do wybierania komunikatu zwracanego przez wywołanie MQGET. Aplikacja ustawia wymagane opcje w tym polu, a następnie ustawia odpowiednie pola w parametrze *MsgDesc* na wartości wymagane dla tych pól. Tylko komunikaty, które mają te wartości w strukturze MQMD dla komunikatu, są kandydatami do pobrania przy użyciu tego parametru *MsgDesc* w wywołaniu MQGET. Pola, dla których odpowiednia opcja zgodności *nie* jest określona, są ignorowane podczas wybierania komunikatu, który ma zostać

zwrócony. Jeśli w wywołaniu MQGET nie określono żadnych kryteriów wyboru (to znaczy *any* komunikat jest akceptowalny), należy ustawić wartość *MatchOptions* na wartość MQMO_NONE.

- W systemie z/OS kryteria wyboru, które mogą być używane, mogą być ograniczone przez typ indeksu używanego w kolejce. Szczegółowe informacje znajdują się w atrybucie kolejki *IndexType*.

Jeśli zostanie określona wartość MQGMO_LOGICAL_ORDER, tylko niektóre komunikaty będą się kwalifikować do powrotu przez następne wywołanie MQGET:

- Jeśli nie istnieje żadna bieżąca grupa lub komunikat logiczny, do zwrotu kwalifikują się tylko komunikaty, które mają *MsgSeqNumber* równe 1 i *Offset* równe 0. W takiej sytuacji można użyć jednej lub kilku spośród następujących opcji dopasowania, aby wybrać, które z zakwalifikowanych komunikatów są zwracane:
 - MQMO_MATCH_MSG_ID
 - MQMO_MATCH_KORELID
 - MQMO_MATCH_GROUP_ID
- Jeśli *jest* bieżąca grupa lub komunikat logiczny, to do zwrotu kwalifikuje się tylko następny komunikat w grupie lub następnym segmencie w komunikacie logicznym, który nie może zostać zmieniony przez podanie opcji MQMO_*.

W obu poprzednich przypadkach można określić opcje zgodności, które nie mają zastosowania, ale wartość odpowiedniego pola w parametrze *MsgDesc* musi być zgodna z wartością odpowiedniego pola w komunikacie, które ma zostać zwrócone; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MATCH_OPTIONS_ERROR jest to warunek, który nie jest spełniony.

Parametr *MatchOptions* jest ignorowany, jeśli zostanie określony parametr MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR.

Pobieranie komunikatów na podstawie właściwości komunikatu nie jest wykonywane przy użyciu opcji zgodności. Więcej informacji na ten temat zawiera sekcja [“SelectionString \(MQCHARV\)” na stronie 489](#).

Można określić jedną lub więcej spośród następujących opcji dopasowania:

MQMO_MATCH_MSG_ID

Komunikat, który ma zostać pobrany, musi mieć identyfikator komunikatu, który jest zgodny z wartością pola *MsgId* w parametrze *MsgDesc* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowaniami, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja zostanie pominięta, pole *MsgId* w parametrze *MsgDesc* zostanie zignorowane, a każdy identyfikator komunikatu będzie zgodny.

Uwaga: Identyfikator komunikatu MQMI_NONE jest wartością specjalną, która jest zgodna z dowolnym identyfikatorem komunikatu w deskryptywie MQMD dla komunikatu. Dlatego podanie parametru MQMO_MATCH_MSG_ID z parametrem MQMI_NONE jest takie samo, jak *nie* określenie wartości MQMO_MATCH_MSG_ID.

MQMO_MATCH_KORELID

Komunikat, który ma zostać pobrany, musi mieć identyfikator korelacji, który jest zgodny z wartością pola *CorrelId* w parametrze *MsgDesc* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowaniami, które mogą mieć zastosowanie (na przykład identyfikator komunikatu).

Jeśli ta opcja zostanie pominięta, pole *CorrelId* w parametrze *MsgDesc* zostanie zignorowane, a dowolny identyfikator korelacji zostanie dopasowany.

Uwaga: Identyfikator korelacji MQCI_NONE jest specjalną wartością, która jest zgodna z dowolnym identyfikatorem korelacji w strukturze MQMD dla komunikatu. Dlatego podanie parametru MQMO_MATCH_CORREL_ID z parametrem MQCI_NONE jest takie samo, jak *nie* z określeniem parametru MQMO_MATCH_CORREL_ID.

MQMO_MATCH_GROUP_ID

Komunikat, który ma zostać pobrany, musi mieć identyfikator grupy, który jest zgodny z wartością pola *GroupId* w parametrze *MsgDesc* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowaniami, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja zostanie pominięta, pole *GroupId* w parametrze *MsgDesc* zostanie zignorowane, a każdy identyfikator grupy będzie zgodny.

Uwaga: Identyfikator grupy MQGI_NONE jest specjalną wartością, która jest zgodna z dowolnym identyfikatorem grupy w strukturze MQMD dla komunikatu. Dlatego podanie wartości MQMO_MATCH_GROUP_ID z parametrem MQGI_NONE jest takie samo, jak wartość *not*, która określa wartość MQMO_MATCH_GROUP_ID.

MQMO_MATCH_MSG_SEQ_NUMBER

Komunikat, który ma zostać pobrany, musi mieć numer kolejny komunikatu, który jest zgodny z wartością pola *MsgSeqNumber* w parametrze *MsgDesc* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład identyfikator grupy).

Jeśli ta opcja zostanie pominięta, pole *MsgSeqNumber* w parametrze *MsgDesc* zostanie zignorowane, a dowolny numer kolejny komunikatu będzie zgodny.

MQMO_MATCH_OFFSET

Komunikat, który ma zostać pobrany, musi mieć przesunięcie zgodne z wartością pola *Offset* w parametrze *MsgDesc* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład numer kolejny komunikatu).

Jeśli ta opcja nie zostanie podana, pole *Offset* w parametrze *MsgDesc* zostanie zignorowane, a wszystkie przesunięcia będą zgodne.

- Ta opcja nie jest obsługiwana w systemie z/OS.

MQMO_MATCH_MSG_TOKEN

Komunikat, który ma zostać pobrany, musi mieć znacznik komunikatu zgodny z wartością pola *MsgToken* w strukturze MQGMO określonej w wywołaniu MQGET.

Tę opcję można określić dla wszystkich kolejek lokalnych. Jeśli zostanie ona określona dla kolejki, która ma *IndexType* o wartości MQIT_MSG_TOKEN (kolejka zarządzana przez WLM), nie można określić żadnych innych opcji zgodności z opcją MQMO_MATCH_MSG_TOKEN.

Nie można określić MQMO_MATCH_MSG_TOKEN z MQGMO_WAIT lub MQGMO_SET_SIGNAL. Jeśli aplikacja chce czekać na pojaw się komunikatu w kolejce z parametrem *IndexType* o wartości MQIT_MSG_TOKEN, należy określić wartość MQMO_NONE.

Jeśli ta opcja zostanie pominięta, pole *MsgToken* w produkcie MQGMO zostanie zignorowane, a każdy znacznik komunikatu będzie zgodny.

Jeśli nie zostanie podana żadna z opisanych opcji, można użyć następującej opcji:

MQMO_NONE

W przypadku wybrania komunikatu, który ma zostać zwrócony, nie można używać żadnych dopasowań. Wszystkie komunikaty w kolejce są zakwalifikowane do pobrania (ale podlegają kontroli za pomocą opcji MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE i MQGMO_COMPLETE_MSG).

Dokumentacja programu pomocy MQMO_NONE. Opcja ta nie jest przeznaczona dla żadnej innej opcji MQMO_*, ale jej wartość jest równa zero, dlatego nie można jej wykryć.

To jest pole wejściowe. Wartością początkową tego pola jest MQMO_MATCH_MSG_ID z parametrem MQMO_MATCH_CORREL_ID. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Uwaga: Początkowa wartość pola *MatchOptions* jest zdefiniowana pod kątem zgodności z wcześniejszymi menedżerami kolejek produktu MQSeries. Jednak podczas odczytywania serii komunikatów z kolejki bez użycia kryteriów wyboru, ta wartość początkowa wymaga, aby aplikacja zresetowała pola *MsgId* i *CorrelId* do wartości MQMI_NONE i MQCI_NONE przed każdym wywołaniem MQGET. Należy unikać konieczności resetowania produktów *MsgId* i *CorrelId*, ustawiając parametr *Version* na wartość MQGMO_VERSION_2, a *MatchOptions* na wartość MQMO_NONE.

Informacje pokrewne

Selektory komunikatów w usłudze JMS

MsgHandle (MQHMSG)

Jeśli określono opcję `MQGMO_PROPERTIES_AS_Q_DEF`, a atrybut kolejki produktu `PropertyControl` nie jest ustawiony na wartość `MQPROP_FORCE_MQRFH2`, to jest to uchwyt do komunikatu, który zostanie wypełniony właściwościami komunikatu pobieranego z kolejki. Uchwyt jest tworzony za pomocą wywołania `MQCRTMH`. Wszystkie właściwości, które są już powiązane z uchwytem, zostaną wyczyszczone przed pobraniem komunikatu.

Można również określić następującą wartość:

`MQHM_NONE`

Nie podano uchwytu komunikatu.

Żaden deskryptor komunikatu nie jest wymagany w wywołaniu `MQGET`, jeśli poprawny uchwyt komunikatu jest dostarczany i używany na wyjściu w celu zawierania właściwości komunikatu, deskryptor komunikatu powiązany z uchwytem komunikatu jest używany dla pól wejściowych.

Jeśli deskryptor komunikatu jest określony w wywołaniu `MQGET`, zawsze ma on pierwszeństwo przed deskryptorem komunikatu powiązanym z uchwytem komunikatu.

Jeśli określono wartość `MQGMO_PROPERTIES_FORCE_MQRFH2` lub określono parametr `MQGMO_PROPERTIES_AS_Q_DEF`, a atrybut kolejki produktu `PropertyControl` ma wartość `MQPROP_FORCE_MQRFH2`, wywołanie nie powiedzie się z kodem przyczyny `MQRC_MD_ERROR`, jeśli nie określono parametru deskryptora komunikatu.

W przypadku powrotu z wywołania `MQGET` właściwości i deskryptor komunikatu powiązane z tym uchwytem komunikatu są aktualizowane w celu odzwierciedlenia stanu pobranego komunikatu (a także deskryptora komunikatu, jeśli został on dostarczony w wywołaniu `MQGET`). Właściwości komunikatu można następnie dowiedzieć się za pomocą wywołania `MQINQMP`.

Poza rozszerzeniami deskryptora komunikatu, jeśli istnieje, właściwość, która może zostać zapytana za pomocą wywołania `MQINQMP`, nie jest zawarta w danych komunikatu. Jeśli komunikat w kolejce zawiera właściwości w danych komunikatu, są one usuwane z danych komunikatu przed zwróceniem danych do aplikacji.

Jeśli żaden uchwyt komunikatu nie jest udostępniony lub wersja jest mniejsza niż `MQGMO_VERSION_4`, należy podać poprawny deskryptor komunikatu w wywołaniu `MQGET`. Wszystkie właściwości komunikatu (z wyjątkiem tych, które znajdują się w deskrytorze komunikatu) są zwracane w danych komunikatu z uwzględnieniem wartości opcji właściwości w strukturze `MQGMO` i atrybutu kolejki produktu `PropertyControl`.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest `MQHM_NONE`. To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż `MQGMO_VERSION_4`.

MsgToken (MQBYTE16)

Pole `MsgToken` -struktura `MQGMO`. To pole jest używane przez menedżer kolejek w celu jednoznacznego zidentyfikowania komunikatu.

Jest to łańcuch bajtów, który jest generowany przez menedżer kolejek w celu jednoznacznego zidentyfikowania komunikatu w kolejce. Znacznik komunikatu jest generowany, gdy komunikat jest najpierw umieszczany w menedżerze kolejek i pozostaje z komunikatem do momentu, gdy komunikat zostanie trwale usunięty z menedżera kolejek, chyba że menedżer kolejek zostanie zrestartowany.

Po usunięciu komunikatu z kolejki produkt `MsgToken`, który zidentyfikował tę instancję komunikatu, nie jest już poprawny i nigdy nie jest ponownie wykorzystywany. Po zrestartowaniu menedżera kolejek produkt `MsgToken`, który zidentyfikował komunikat w kolejce przed restartowaniem, może nie być poprawny po restarcie. Jednak `MsgToken` nigdy nie jest ponownie wykorzystywana do identyfikowania innej instancji komunikatu. `MsgToken` jest generowany przez menedżer kolejek i nie jest widoczny dla żadnej aplikacji zewnętrznej.

Gdy komunikat jest zwracany przez wywołanie MQGET, w którym podano wersję 3 lub wyższą wartość MQGMO, program *MsgToken* identyfikujący komunikat w kolejce jest zwracany przez menedżer kolejek w produkcie MQGMO. Wystąpił jeden wyjątek: jeśli komunikat jest usuwany z kolejki poza punktem synchronizacji, menedżer kolejek może nie zwrócić *MsgToken*, ponieważ nie jest on przydatny do identyfikowania zwróconego komunikatu w kolejnych wywołaniach MQGET. Aplikacje powinny używać produktu *MsgToken* tylko do odwołania się do komunikatu w kolejnych wywołaniach MQGET.

Jeśli podano *MsgToken* i określono parametr *MatchOption* MQMO_MATCH_MSG_TOKEN, a nie określono wartości MQGMO_MSG_UNDER_CURSOR ani MQGMO_BROWSE_MSG_UNDER_CURSOR, wówczas można zwrócić tylko komunikat identyfikowany przez produkt *MsgToken*. Opcja ta jest poprawna we wszystkich kolejkach lokalnych niezależnie od wartości INDXTYPE, a w systemie z/OS należy używać parametru INDXTYPE (MSGTOKEN) tylko w kolejkach menedżera obciążenia (WLM).

Wszystkie inne określone *MatchOptions* są zaznaczone, a jeśli nie są one zgodne, zwracana jest wartość MQRC_NO_MSG_AVAILABLE. Jeśli parametr MQGMO_BROWSE_NEXT jest kodowany za pomocą komendy MQMO_MATCH_MSG_TOKEN, komunikat identyfikowany przez składnik *MsgToken* jest zwracany tylko wtedy, gdy znajduje się poza kursorem przeglądania dla uchwytu wywołującego.

Jeśli określono wartość MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR, wartość MQMO_MATCH_MSG_TOKEN jest ignorowana.

Parametr MQMO_MATCH_MSG_TOKEN nie jest poprawny z następującymi opcjami pobierania komunikatów:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

W przypadku wywołania MQGET określającego wartość MQMO_MATCH_MSG_TOKEN należy podać wartość MQGMO w wersji 3 lub nowszej do wywołania, w przeciwnym razie zwracana jest wartość MQRC_WRONG_GMO_VERSION.

Jeśli *MsgToken* nie jest w tej chwili poprawna, zwracana jest wartość MQCC_FAILED z MQRC_NO_MSG_AVAILABLE, chyba że wystąpił inny błąd.

Opcje (MQLONG)

Opcje produktu **MQGMO** sterują działaniem produktu MQGET. Możliwe jest określenie wartości zero lub większej liczby opcji. Jeśli wymagana jest więcej niż jedna wartość opcjonalna:

- Dodaj wartości (nie należy dodawać tej samej stałej więcej niż raz), lub
- Połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Kombinacje opcji, które nie są poprawne, są oznaczane; wszystkie pozostałe kombinacje są poprawne.

Opcje oczekiwania: Następujące opcje odnoszą się do oczekiwania na przybycie komunikatów do kolejki:

MQGMO_WAIT

Aplikacja czeka, aż pojawi się odpowiedni komunikat. Maksymalny czas, przez jaki aplikacja oczekuje na podanie w *WaitInterval*.

Ważne: Nie ma oczekiwania lub opóźnienia, jeśli odpowiedni komunikat jest dostępny natychmiast.

Jeśli żądania MQGET są zablokowane lub żądania MQGET stają się blokowane podczas oczekiwania, to oczekiwanie zostanie anulowane. Wywołanie kończy się MQCC_FAILED i kodem przyczyny MQRC_GET_INHIBITED, niezależnie od tego, czy w kolejce znajdują się odpowiednie komunikaty.

Programu MQGMO_WAIT można używać z opcjami MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Jeśli kilka aplikacji oczekuje w tej samej kolejce współużytkowanej, wówczas następujące reguły wybierają, która aplikacja jest aktywowana po nadejściu odpowiedniego komunikatu:

Tabela 45. Reguły aktywowania MQGET wywołań w kolejce współużytkowanej.		
Liczba wywołań MQGET oczekujących na aktywację		Wynik
Za pomocą opcji BROWSE	Bez opcji BROWSE ¹	
Brak	jeden lub więcej	Jedno wywołanie MQGET bez opcji BROWSE jest aktywowane.
jeden lub więcej	Brak	Wszystkie wywołania MQGET z opcją BROWSE są aktywowane.
jeden lub więcej	jeden lub więcej	Jedno wywołanie MQGET bez opcji BROWSE jest aktywowane. Liczba aktywowanych wywołań MQGET z aktywowaną opcją BROWSE jest nieprzewidywalna.

Jeśli więcej niż jedno wywołanie MQGET bez opcji BROWSE oczekuje w tej samej kolejce, zostanie aktywowana tylko jedna kolejka. Menedżer kolejek próbuje nadać priorytet oczekującym w następującej kolejności:

1. Konkretne żądania get-wait, które mogą być spełnione tylko przez niektóre komunikaty, na przykład takie, które mają konkretną wartość *MsgId* lub *CorrelId* (lub obie te wartości).
2. Ogólne żądania pobierania, które mogą być spełnione przez dowolny komunikat.

Uwaga:

- W ramach pierwszej kategorii nie nadano dodatkowego priorytetu dla bardziej konkretnych żądań pobierania. Na przykład żądania, które określają zarówno *MsgId*, jak i *CorrelId*.
- W ramach jednej z kategorii nie można przewidzieć, która aplikacja jest wybrana. W szczególności, najdłuższy czas oczekiwania aplikacji nie musi być wybrany.
- Długość ścieżki oraz uwagi dotyczące planowania priorytetów w systemie operacyjnym mogą oznaczać, że aplikacja oczekująca o niższym priorytecie systemu operacyjnego niż oczekiwano pobiera komunikat.
- Może się również zdarzyć, że aplikacja, która nie oczekuje na pobranie komunikatu, będzie preferowana względem tego, który jest.

W systemie z/OS mają zastosowanie następujące punkty:

- Jeśli aplikacja ma kontynuować pracę z innymi pracami podczas oczekiwania na nadejście komunikatu, należy rozważyć użycie opcji sygnalizacji (MQGMO_SET_SIGNAL). Jednak opcja sygnału jest specyfika dla środowiska; aplikacje, które należy do portów między różnymi środowiskami, nie mogą być używane.
- Jeśli dla tego samego komunikatu oczekuje więcej niż jedno wywołanie MQGET, z mieszaniną opcji oczekiwania i sygnału, każde oczekujące wywołanie jest traktowane jednakowo. Jest to błąd, który określa MQGMO_SET_SIGNAL przy użyciu produktu MQGMO_WAIT. Podanie tej opcji z uchwycem kolejki, dla którego sygnał jest wybitny, jest również błędem.
- If you specify MQGMO_WAIT or MQGMO_SET_SIGNAL for a queue that has an *IndexType* of MQIT_MSG_TOKEN, no selection criteria are permitted. Oznacza to, że:
 - Jeśli używana jest wersja version-1 MQGMO, należy ustawić pola *MsgId* i *CorrelId* w polu MQMD określonym w wywołaniu MQGET na MQMI_NONE i MQCI_NONE.
 - W przypadku korzystania z produktu MQGMO w wersji version-2 lub nowszej należy ustawić pole *MatchOptions* na wartość MQMO_NONE.
- W przypadku wywołania MQGET w kolejce współużytkowanej, gdy wywołanie jest żądaniem przeglądania lub destruktywnym plikiem komunikatu grupy, a ani *MsgId*, ani *CorrelId* nie

¹ Wywołanie MQGET określające opcję MQGMO_LOCK jest traktowane jako wywołanie bez przeglądania.

mają być dopasowywane, sygnał EBC jest umieszczany w tabeli MQEC_MSG_PRZYBYŁ po 200 milisekund.

Dzieje się tak, nawet jeśli odpowiedni komunikat mógł nie zostać wystany do kolejki, dopóki czas oczekiwania nie upłynie, gdy kolejka jest delegowana z opcją MQEC_WAIT_INTERVAL_EXPIRED. Po wystaniu komendy MQEC_MSG_PRZYBYŁ należy ponownie wydać drugie wywołanie programu MQGET w celu pobrania komunikatu, jeśli jest on dostępny.

Ta technika jest używana w celu zapewnienia, że użytkownik jest informowany w odpowiednim czasie o przybyciu komunikatu, ale może być wyświetlany jako niespodziewany narzut przetwarzania w porównaniu z podobną sekwencją wywołań w kolejce niewspółużytkowanej.

MQGMO_WAIT jest ignorowany, jeśli jest określony za pomocą opcji MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_MSG_UNDER_CURSOR ; nie jest zgłaszany żaden błąd.

MQGMO_NO_WAIT

Aplikacja nie czeka, jeśli nie będzie dostępny żaden odpowiedni komunikat. MQGMO_NO_WAIT jest przeciwieństwem MQGMO_WAIT. MQGMO_NO_WAIT jest zdefiniowana w dokumentacji programu pomocowego. Jest to wartość domyślna, jeśli nie zostanie podana żadna wartość.

MQGMO_SET_SIGNAL

Tej opcji należy użyć razem z polami *Signal1* i *Signal2*. Umożliwia on aplikacjom kontynuowanie pracy z innymi pracami podczas oczekiwania na nadejście komunikatu. Umożliwia on także (jeśli dostępne są odpowiednie urządzenia systemu operacyjnego) aplikacje, które oczekują na komunikaty przychodzące do więcej niż jednej kolejki.

Uwaga: Opcja MQGMO_SET_SIGNAL jest specyfika dla środowiska; nie należy używać jej dla aplikacji, które mają zostać użyte do portu.

W dwóch przypadkach wywołanie kończy się w taki sam sposób, jak w przypadku, gdy nie określono tej opcji:

1. Jeśli aktualnie dostępny komunikat spełnia kryteria określone w deskrytorze komunikatu.
2. Jeśli wykryty zostanie błąd parametru lub inny błąd synchroniczny,

Jeśli żaden komunikat spełniający kryteria określone w deskrytorze komunikatu jest aktualnie dostępny, sterowanie powraca do aplikacji bez czekania na komunikat, który ma zostać dostarczony. Parametry *CompCode* i *Reason* są ustawione na wartość MQCC_WARNING i MQRC_SIGNAL_REQUEST_ACCEPTED. Inne pola wyjściowe w deskrytorze komunikatu i parametry wyjściowe wywołania MQGET nie są ustawione. Po nadejściu stosownego komunikatu, sygnał jest dostarczany poprzez opublikowanie EBC.

Program wywołujący musi następnie ponownie wywołać wywołanie MQGET w celu pobrania komunikatu. Aplikacja może czekać na ten sygnał, korzystając z funkcji udostępnianych przez system operacyjny.

Jeśli system operacyjny udostępnia mechanizm wielokrotnego oczekiwania, można go użyć w celu oczekiwania na przestanie komunikatu do dowolnej z kilku kolejek.

Jeśli określono niezerową wartość *WaitInterval*, sygnał jest dostarczany po upływie okresu oczekiwania. Menedżer kolejek może również anulować oczekiwanie, w którym to przypadku sygnał jest dostarczany.

Więcej niż jedno wywołanie MQGET może ustawić sygnał dla tego samego komunikatu. Kolejność, w jakiej aplikacje są aktywowane, jest taka sama, jak opisana w sekcji MQGMO_WAIT.

Jeśli więcej niż jedno wywołanie MQGET oczekuje na ten sam komunikat, każde oczekujące wywołanie jest traktowane jednakowo. Połączenia mogą zawierać kombinację opcji oczekiwania i sygnału.

W pewnych warunkach wywołanie MQGET może pobrać komunikat, a sygnał wynikający z przybycia tego samego komunikatu może zostać dostarczony. Po dostarczeniu sygnału musi być przygotowany wniosek, aby żaden komunikat nie był dostępny.

Uchwyty kolejki nie może zawierać więcej niż jednego oczekiwającego żądania sygnału.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_UNLOCK
- MQGMO_WAIT

W przypadku wywołania MQGET w kolejce współużytkowanej, gdy wywołanie jest żądaniem przeglądania, lub destrukcyjnym dostaniem komunikatu grupowego, a ani *MsgId*, ani *CorrelId* nie mają być dopasowane, sygnał użytkownika EBC jest księgowany MQEC_MSG_ARRIVED po 200 milisekund.

Dzieje się tak, mimo że odpowiedni komunikat mógł nie zostać umieszczony w kolejce, dopóki nie upłynął okres oczekiwania, gdy kolejka jest publikowanej z produktem MQEC_WAIT_INTERVAL_EXPIRED. Po wysłaniu produktu MQEC_MSG_ARRIVED należy ponownie wydać drugie wywołanie programu MQGET w celu pobrania komunikatu, jeśli jest on dostępny.

Ta technika jest używana w celu zapewnienia, że użytkownik jest informowany w odpowiednim czasie o przybyciu komunikatu, ale może być wyświetlany jako niespodziewany narzut przetwarzania w porównaniu z podobną sekwencją wywołań w kolejce niewspółużytkowanej.

Nie jest to wydajna metoda pobierania komunikatów, gdy komunikaty są dodawane rzadko. Aby uniknąć tego narzutu dla przypadku przeglądania, należy określić wartość *MsgId* (jeśli nie jest indeksowana lub poindeksowana przez produkt *MsgId*) lub *CorrelId* (jeśli jest indeksowana przez *CorrelId*), która jest zgodna z wywołaniem MQGET.

Ta opcja jest obsługiwana tylko w systemie z/OS.

MQGMO_FAIL_IF QUIESCING

Wymusi niepowodzenie wywołania MQGET, jeśli menedżer kolejek znajduje się w stanie wygaszania.

W systemie z/OS ta opcja również wymusza niepowodzenie wywołania MQGET, jeśli połączenie (dla aplikacji CICS lub IMS) znajduje się w stanie wygaszania.

Jeśli ta opcja jest określona za pomocą opcji MQGMO_WAIT lub MQGMO_SET_SIGNAL, a czas oczekiwania lub sygnał jest zaległy w momencie, gdy menedżer kolejek przechodzi do stanu wygaszania:

- Oczekiwanie zostało anulowane, a wywołanie zwraca kod zakończenia MQCC_FAILED z kodem przyczyny MQRC_Q_MGR QUIESCING lub MQRC_CONNECTION QUIESCING.
- Sygnał jest anulowany ze specyficznym dla środowiska kodem zakończenia sygnału.

W systemie z/OS sygnał kończy się kodem zakończenia zdarzenia MQEC_Q_MGR QUIESCING lub MQEC_CONNECTION QUIESCING.

Jeśli parametr MQGMO_FAIL_IF QUIESCING nie zostanie określony, a menedżer kolejek lub połączenie zostanie wprowadzone do stanu wygaszania, to oczekiwanie lub sygnał nie zostanie anulowany.

Opcje punktu synchronizacji: Następujące opcje odnoszą się do udziału wywołania MQGET w jednostce pracy:

MQGMO_SYNCPOINT

Żądanie ma działać w ramach normalnych protokołów jednostkowych pracy. Komunikat jest oznaczony jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzana. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy jest wycofana.

Można pozostawić ustawienie MQGMO_SYNCPOINT i MQGMO_NO_SYNCPOINT nie ustawione. W takim przypadku włączenie żądania pobrania w protokołach jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek. Nie jest on określany przez środowisko, na którym działa aplikacja. W systemie z/OS żądanie pobrania znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie pobrania nie znajduje się w obrębie jednostki pracy.

Z powodu tych różnic aplikacja, której chcesz użyć do portu, nie może zezwalać na ustawienie domyślne tej opcji; należy jawnie określić wartość MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_SYNCPOINT_IF_PERSISTENT

Żądanie ma działać w normalnych protokołach jednostki pracy, ale *tylko* , jeśli odczyta wiadomość jest trwała. Komunikat trwały ma wartość MQPER_PERSISTENT w polu *Persistence* w MQMD.

- Jeśli komunikat jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja określiła wartość MQGMO_SYNCPOINT.
- Jeśli komunikat nie jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja określiła wartość MQGMO_NO_SYNCPOINT.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_COMPLETE_MSG
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT
- MQGMO_UNLOCK

Ta opcja jest obsługiwana w następujących środowiskach: AIX, HP-UX, z/OS, IBM i, Solaris i Linux oraz IBM MQ MQI clients połączone z tymi systemami.

MQGMO_NO_SYNCPOINT

Wniosek ma działać poza normalnymi protokołami jednostki pracy. Jeśli zostanie wyświetlony komunikat bez opcji przeglądania, zostanie on natychmiast usunięty z kolejki. Komunikat nie może zostać ponownie udostępniony przez utworzenie kopii zapasowej jednostki pracy.

Ta opcja jest przyjmowana w przypadku określenia wartości MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Można pozostawić ustawienie MQGMO_SYNCPOINT i MQGMO_NO_SYNCPOINT nie ustawione. W takim przypadku włączenie żądania pobrania w protokołach jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek. Nie jest on określany przez środowisko, na którym działa aplikacja. W systemie z/OS żądanie pobrania znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie pobrania nie znajduje się w obrębie jednostki pracy.

Z powodu tych różnic aplikacja, której chcesz użyć do portu, nie może zezwalać na ustawienie domyślne tej opcji; należy jawnie określić wartość MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT .

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT

Utwórz kopię zapasową jednostki pracy bez przywrócenia w kolejce wiadomości, która została oznaczona przy użyciu tej opcji.

Ta opcja jest obsługiwana tylko w systemie z/OS.

Jeśli ta opcja jest określona, należy również określić wartość MQGMO_SYNCPOINT .
MQGMO_MARK_SKIP_BACKOUT nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Uwaga: W systemach IMS i CICS może być konieczne wydanie ekstraklasy IBM MQ po utworzeniu kopii zapasowej jednostki pracy zawierającej komunikat oznaczony MQGMO_MARK_SKIP_BACKOUT. Przed zatwierdzeniem nowej jednostki pracy zawierającej oznaczoną wiadomość należy wywołać wywołanie IBM MQ . Wywołanie może być dowolne IBM MQ , jak chcesz.

1. On IMS, if you have not applied IMS APAR PN60855 and you are running an IMS MPP or BMP application.
2. W systemie CICS, jeśli uruchomiona jest dowolna aplikacja.

W obu przypadkach przed zatwierdzeniem nowej jednostki pracy zawierającej wycofany komunikat należy wprowadzić wywołanie IBM MQ .

Uwaga: W ramach jednostki pracy może istnieć tylko jedno żądanie pobrania oznaczone jako pominięcie wycofania, a także brak lub kilka nieoznaczonych żądań pobierania.

Jeśli aplikacja wycofuje się z jednostki pracy, komunikat, który został pobrany za pomocą programu MQGMO_MARK_SKIP_BACKOUT , nie zostanie odtworzony do poprzedniego stanu. Wycofuje się inne aktualizacje zasobów. Komunikat jest traktowany tak, jakby został pobrany w nowej jednostce pracy uruchomionej przez żądanie wycofania. Komunikat jest pobierany bez opcji MQGMO_MARK_SKIP_BACKOUT .

MQGMO_MARK_SKIP_BACKOUT jest przydatna, jeśli po zmianie niektórych zasobów staje się oczywiste, że jednostka pracy nie może zakończyć się pomyślnie. Jeśli ta opcja zostanie pominięta, utworzenie kopii zapasowej jednostki pracy spowoduje przywrócenie komunikatu w kolejce. Ta sama sekwencja zdarzeń pojawia się ponownie, gdy komunikat jest pobierany.

Jeśli jednak w oryginalnym wywołaniu programu MQGET zostanie określona wartość MQGMO_MARK_SKIP_BACKOUT , to podczas tworzenia kopii zapasowej jednostki pracy wycofuje się aktualizacje pozostałych zasobów. Komunikat jest traktowany tak, jakby został pobrany w ramach nowej jednostki pracy. Aplikacja może wykonać odpowiednią obsługę błędów. Może on wysłać komunikat raportu do nadawcy oryginalnego komunikatu lub umieścić oryginalny komunikat w kolejce niedostarczonych komunikatów. Następnie może zatwierdzić nową jednostkę pracy. Zatwierdzenie nowej jednostki pracy powoduje trwałe usunięcie komunikatu z oryginalnej kolejki.

MQGMO_MARK_SKIP_BACKOUT oznacza pojedynczy komunikat fizyczny. Jeśli komunikat należy do grupy komunikatów, inne komunikaty w grupie nie są oznaczane. Podobnie, jeśli zaznaczony komunikat jest segmentem komunikatu logicznego, pozostałe segmenty w komunikacie logicznym nie są oznaczone.

Każdy komunikat w grupie może być oznaczony, ale jeśli komunikaty są pobierane za pomocą MQGMO_LOGICAL_ORDER, to korzystne jest zaznaczenie pierwszego komunikatu w grupie. Jeśli zostanie utworzona kopia zapasowa jednostki pracy, pierwsza (oznaczona) wiadomość zostanie przeniesiona do nowej jednostki pracy. Drugi i późniejszy komunikat w grupie jest przywrócony do kolejki. Komunikaty pozostawione w kolejce nie mogą być pobierane przez inną aplikację przy użyciu programu MQGMO_LOGICAL_ORDER. Pierwszy komunikat w grupie nie znajduje się już w kolejce. Jednak aplikacja, która bazuje na jednostce pracy, może pobrać drugie i późniejsze komunikaty do nowej jednostki pracy przy użyciu opcji MQGMO_LOGICAL_ORDER . Pierwszy komunikat został już pobrany.

Sporadycznie może być konieczne utworzenie kopii zapasowej nowej jednostki pracy. Na przykład, ponieważ kolejka niedostarczonych komunikatów jest pełna, a komunikat nie może być odrzucony. Utworzenie kopii zapasowej nowej jednostki pracy powoduje przywrócenie komunikatu w oryginalnej kolejce, co uniemożliwia utratę komunikatu. Jednak w tym przypadku przetwarzanie nie może być kontynuowane. Po utworzeniu kopii zapasowej nowej jednostki pracy, aplikacja musi poinformować operatora lub administratora o tym, że wystąpił nienaprawialny błąd, a następnie trzeba zakończyć pracę.

Produkt MQGMO_MARK_SKIP_BACKOUT działa tylko wtedy, gdy jednostka pracy zawierająca żądanie pobrania zostanie przerwana przez aplikację, która je wycofa. Jeśli zostanie utworzona kopia zapasowa jednostki pracy zawierającej żądanie pobrania, ponieważ transakcja lub system nie powiedzie się, program MQGMO_MARK_SKIP_BACKOUT zostanie zignorowany. Każdy komunikat pobrany za pomocą tej opcji jest ponownie przywrócony do kolejki w taki sam sposób, jak komunikaty pobierane bez tej opcji.

Opcje przeglądania: Następujące opcje są powiązane z przeglądaniem komunikatów w kolejce:

MQGMO_BROWSE_FIRST

Gdy kolejka jest otwierana za pomocą opcji MQOO_BROWSE, kursor przeglądania jest ustanawiany, umieszczany logicznie przed pierwszym komunikatem w kolejce. Następnie można użyć wywołań MQGET , podając opcję MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT lub MQGMO_BROWSE_MSG_UNDER_CURSOR , aby pobrać komunikaty z kolejki bez zniszczenia. Kursor przeglądania oznacza pozycję w obrębie komunikatów znajdujących się w kolejce, z której następne wywołanie MQGET z MQGMO_BROWSE_NEXT wyszukuje odpowiedni komunikat.

MQGMO_BROWSE_FIRST nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Wywołanie MQGET z programem MQGMO_BROWSE_FIRST ignoruje poprzednią pozycję kursora przeglądania. Wczytany jest pierwszy komunikat w kolejce, który spełnia warunki określone w deskrytorze komunikatu. Komunikat pozostaje w kolejce, a kursor przeglądania znajduje się w tym komunikacie.

Po wywołaniu tej operacji kursor przeglądania jest ustawiony na zwróconej wiadomości. Komunikat może zostać usunięty z kolejki, zanim zostanie wydane następne wywołanie MQGET z MQGMO_BROWSE_NEXT . W tym przypadku kursor przeglądania pozostaje w kolejce zajmowanej przez komunikat, mimo że pozycja ta jest teraz pusta.

Aby usunąć komunikat z kolejki, należy użyć opcji MQGMO_MSG_UNDER_CURSOR z wywołaniem MQGET bez przeglądania.

Kursor przeglądania nie jest przenoszony za pomocą wywołania MQGET bez przeglądania, nawet jeśli używany jest ten sam uchwyt *Hobj* . Nie jest też przenoszony przez przeglądanie MQGET , które zwraca kod zakończenia MQCC_FAILED, lub kod przyczyny MQRC_TRUNCATED_MSG_FAILED.

W celu zablokowania przeglądanych komunikatów należy określić opcję MQGMO_LOCK (z tą opcją).

Użytkownik może określić MQGMO_BROWSE_FIRST z dowolną poprawną kombinacją opcji MQGMO_* i MQMO_* , które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli zostanie określona wartość MQGMO_LOGICAL_ORDER, komunikaty są przeglądane w kolejności logicznej. Pominięcie tej opcji powoduje, że komunikaty są przeglądane w porządku fizycznym.

Jeśli zostanie określona opcja MQGMO_BROWSE_FIRST, można przetaczać się między porządkiem logicznym a porządkiem fizycznym. Kolejne wywołania programu MQGET przy użyciu programu MQGMO_BROWSE_NEXT przeglądną kolejkę w tej samej kolejności, co najnowsze wywołanie, które określono MQGMO_BROWSE_FIRST dla uchwytu kolejki.

Menedżer kolejek zachowuje dwa zestawy informacji o grupach i segmentach dla wywołań programu MQGET. Informacje o grupach i segmentach dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki. Jeśli zostanie określona wartość MQGMO_BROWSE_FIRST, menedżer kolejek zignoruje informacje o grupie i segmencie w celu ich przeglądania. Skanuje on kolejkę tak, jakby nie było żadnej bieżącej grupy i nie ma bieżącego komunikatu logicznego. Jeśli wywołanie MQGET powiedzie się, kod zakończenia MQCC_OK lub MQCC_WARNING, informacje o grupie i segmencie dla przeglądania zostaną ustawione na wartość zwróconego komunikatu. Jeśli wywołanie nie powiedzie się, informacje o grupie i segmencie pozostaną takie same, jak przed wywołaniem.

MQGMO_BROWSE_NEXT

Przejdź do następnego komunikatu w kolejce, który spełnia kryteria wyboru określone w wywołaniu programu MQGET, przejdź do następnego komunikatu. Komunikat jest zwracany do aplikacji, ale pozostaje w kolejce.

MQGMO_BROWSE_NEXT nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Produkt MQGMO_BROWSE_NEXT działa w taki sam sposób, jak produkt MQGMO_BROWSE_FIRST, jeśli jest to pierwsze wywołanie do przeglądania kolejki po otwarciu kolejki w celu jego przeglądania.

Komunikat znajdujący się pod kursorem może zostać usunięty z kolejki, zanim zostanie wydane następne wywołanie MQGET z MQGMO_BROWSE_NEXT. Kursor przeglądania pozostaje logicznie na pozycji w kolejce, którą zajmował komunikat, mimo że pozycja ta jest teraz pusta.

Komunikaty są zapisywane w kolejce na jeden z dwóch sposobów:

- FIFO w ramach priorytetu (MQMDS_PRIORITY), lub
- FIFO *niezależnie* od priorytetu (MQMDS_FIFO)

Atrybut kolejki *MsgDeliverySequence* wskazuje, która metoda ma zastosowanie (szczegółowe informacje na ten temat zawiera sekcja [“Atrybuty dla kolejek” na stronie 845](#)).

Kolejka może mieć *MsgDeliverySequence* z MQMDS_PRIORITY. Komunikat dociera do kolejki o wyższym priorytecie niż ten, w którym znajduje się kursor, na którym aktualnie znajduje się kursor przeglądania. W takim przypadku komunikat o wyższym priorytecie nie zostanie znaleziony podczas bieżącego przeglądania kolejki przy użyciu programu MQGMO_BROWSE_NEXT. Można ją znaleźć tylko po zresetowaniu kursora za pomocą opcji MQGMO_BROWSE_FIRST lub ponownym otwarciu kolejki.

Opcja MQGMO_MSG_UNDER_CURSOR może być używana z wywołaniem MQGET bez przeglądania, jeśli jest to wymagane, w celu usunięcia komunikatu z kolejki.

Kursor przeglądania nie jest przenoszony przez wywołania MQGET bez przeglądania przy użyciu tego samego uchwytu *Hobj*.

Aby zablokować przejrzenie komunikatu, należy określić opcję MQGMO_LOCK z tą opcją.

Użytkownik może określić MQGMO_BROWSE_NEXT z dowolną poprawną kombinacją opcji MQGMO_* i MQMO_*, które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli zostanie określona wartość MQGMO_LOGICAL_ORDER, komunikaty są przeglądane w kolejności logicznej. Pominięcie tej opcji powoduje, że komunikaty są przeglądane w porządku fizycznym. Jeśli zostanie określona opcja MQGMO_BROWSE_FIRST, można przełączać się między porządkiem logicznym a porządkiem fizycznym. Kolejne wywołania programu MQGET przy użyciu programu MQGMO_BROWSE_NEXT przeglądają kolejkę w tej samej kolejności, co najnowsze wywołanie, które określono MQGMO_BROWSE_FIRST dla uchwytu kolejki. Wywołanie nie powiodło się z kodem przyczyny MQRC_INCONSISTENT_BROWSE, jeśli ten warunek nie jest spełniony.

Uwaga: Jeśli program MQGMO_LOGICAL_ORDER nie został określony, należy zachować szczególną ostrożność podczas korzystania z wywołania MQGET w celu przejrzania poza koniec grupy komunikatów. Na przykład założmy, że ostatni komunikat w grupie poprzedza pierwszy komunikat w grupie w kolejce. Jeśli program MQGMO_BROWSE_NEXT jest używany do przeglądania poza końcem grupy, podanie wartości MQMO_MATCH_MSG_SEQ_NUMBER z parametrem *MsgSeqNumber* ustawionym na wartość 1 spowoduje zwrócenie pierwszego komunikatu w grupie, który już został przejrzany. Ten wynik może wystąpić natychmiast lub liczba wywołań MQGET w późniejszym czasie, jeśli istnieją grupy, które są interweniowane. To samo rozważanie dotyczy komunikatu logicznego, który nie znajduje się w grupie.

Informacje o grupach i segmentach dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki.

MQGMO_BROWSE_MSG_UNDER_CURSOR

Pobranie komunikatu wskazywanego przez kursor przeglądania bez zniszczenia, niezależnie od opcji MQMO_* określonych w polu *MatchOptions* w MQGMO.

MQGMO_BROWSE_MSG_UNDER_CURSOR nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Komunikat wskazywał na kursor przeglądania, który został ostatnio pobrany przy użyciu opcji MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT. Wywołanie nie powiedzie się, jeśli żadna z tych wywołań nie została wywołana dla tej kolejki od momentu jej otwarcia. Wywołanie nie powiedzie się również w przypadku, gdy komunikat, który był pod kursorem przeglądania, został pobrany w sposób destruktywny.

Pozycja kursora przeglądania nie jest zmieniana przez to wywołanie.

Opcja MQGMO_MSG_UNDER_CURSOR może być używana z wywołaniem MQGET bez przeglądania, aby usunąć komunikat z kolejki.

Kursor przeglądania nie jest przenoszony za pomocą wywołania MQGET bez przeglądania, nawet jeśli używany jest ten sam uchwyt *Hobj*. Nie jest też przenoszone przez przeglądanie MQGET, które zwraca kod zakończenia MQCC_FAILED, lub kod przyczyny MQRC_TRUNCATED_MSG_FAILED.

Jeśli MQGMO_BROWSE_MSG_UNDER_CURSOR jest określony za pomocą MQGMO_LOCK:

- Jeśli jest już zablokowany komunikat, musi to być ten, który znajduje się pod kursorem, tak aby został zwrócony bez odblokowania i blokowania. Komunikat pozostaje zablokowany.

- Jeśli nie ma zablokowanego komunikatu i pod kursorem przeglądania znajduje się komunikat, jest on zablokowany i zwracany do aplikacji. Jeśli pod kursorem przeglądania nie ma żadnego komunikatu, wywołanie nie powiedzie się.

Jeśli wartość MQGMO_BROWSE_MSG_UNDER_CURSOR jest określona bez MQGMO_LOCK:

- Jeśli jest już zablokowany komunikat, musi to być ten, który znajduje się pod kursorem. Komunikat zostanie zwrócony do aplikacji, a następnie odblokowany. Ponieważ komunikat jest teraz odblokowany, nie ma gwarancji, że można go ponownie przeglądać lub pobrać destruktywnie przez tę samą aplikację. Być może została ona pobrana w sposób destruktywny przez inną aplikację pobierającą komunikaty z kolejki.
- Jeśli nie ma zablokowanego komunikatu, a pod kursorem przeglądania znajduje się komunikat, jest on zwracany do aplikacji. Jeśli pod kursorem przeglądania nie ma komunikatu, wywołanie nie powiedzie się.

Jeśli parametr MQGMO_COMPLETE_MSG jest określony w produkcie MQGMO_BROWSE_MSG_UNDER_CURSOR, kursor przeglądania musi zidentyfikować komunikat, którego pole *Offset* w MQMD ma wartość zero. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

Informacje o grupach i segmentach dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki.

MQGMO_MSG_UNDER_CURSOR

Pobieranie komunikatu wskazanego przez kursor przeglądania, niezależnie od opcji MQMO_* określonych w polu *MatchOptions* w sekcji MQGMO. Komunikat jest usuwany z kolejki.

Komunikat wskazywał na kursor przeglądania, który został ostatnio pobrany przy użyciu opcji MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Jeśli parametr MQGMO_COMPLETE_MSG jest określony w produkcie MQGMO_MSG_UNDER_CURSOR, kursor przeglądania musi zidentyfikować komunikat, którego pole *Offset* w MQMD ma wartość zero. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta zarówno dla przeglądania, jak i dla danych wejściowych. Jeśli kursor przeglądania nie wskazuje aktualnie odtwarzalnego komunikatu, wywołanie MQGET zwróci błąd.

MQGMO_MARK_BROWSE_HANDLE

Komunikat zwracany przez pomyślny MQGET lub identyfikowany przez zwróconej *MsgToken* jest oznaczony. Znak jest specyficzny dla uchwytu obiektu używanego w wywołaniu.

Komunikat nie został usunięty z kolejki.

MQGMO_MARK_BROWSE_HANDLE jest poprawna tylko wtedy, gdy zostanie określona jedna z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE

- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Komunikat pozostaje w tym stanie do momentu wystąpienia jednego z następujących zdarzeń:

- Rozpatrywany uchwyt obiektu jest zamykany, w normalnych warunkach lub w inny sposób.
- Komunikat nie jest oznaczany dla tego uchwytu przy użyciu wywołania MQGET z opcją MQGMO_UNMARK_BROWSE_HANDLE.
- Komunikat jest zwracany z wywołania do destruktywnego MQGET, który kończy się na MQCC_OK lub MQCC_WARNING. Stan komunikatu pozostaje zmieniony nawet wtedy, gdy MQGET jest później wycofany.
- Komunikat utraci ważność.

MQGMO_MARK_BROWSE_CO_OP

Komunikat zwrócony przez udaną MQGET lub identyfikowany przez zwróconego *MsgToken* jest oznaczony dla wszystkich uchwytów w zestawie współpracującym.

Znacznik poziomu kooperatywnego jest dodatkowo używany do dowolnego znaku poziomu uchwytu, który mógł zostać ustawiony.

Komunikat nie został usunięty z kolejki.

MQGMO_MARK_BROWSE_CO_OP jest poprawny tylko wtedy, gdy używany uchwyt obiektu został zwrócony przez wywołanie do MQOPEN, które określono MQOO_CO_OP. Należy również określić jedną z następujących opcji MQGMO:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Jeśli komunikat jest już oznaczony, a opcja MQGMO_UNMARKED_BROWSE_MSG nie jest określona, wywołanie kończy się niepowodzeniem z MQCC_FAILED i kodem przyczyny MQRC_MSG_MARKED_BROWSE_CO_OP.

Komunikat pozostaje w tym stanie do momentu wystąpienia jednego z następujących zdarzeń:

- Wszystkie uchwyty obiektów w zestawie współpracującym są zamykane.
- Komunikat nie jest oznaczany dla współpracujących przeglądarek przy użyciu wywołania MQGET z opcją MQGMO_UNMARK_BROWSE_CO_OP.
- Komunikat jest automatycznie nieoznaczony przez menedżer kolejek.
- Komunikat jest zwracany z wywołania do MQGET bez przeglądania. Stan komunikatu pozostaje zmieniony nawet wtedy, gdy MQGET jest później wycofany.
- Komunikat utraci ważność.

MQGMO_UNMARKED_BROWSE_MSG

Wywołanie funkcji MQGET, które określa, że program MQGMO_UNMARKED_BROWSE_MSG zwraca komunikat, który jest uważany za nieoznaczony dla uchwytu. Komunikat nie zwraca komunikatu,

jeśli komunikat został oznaczony jako uchwyt. Nie zwraca również komunikatu, jeśli kolejka została otwarta za pomocą wywołania MQOPEN, z opcją MQOO_CO_OP, a komunikat został oznaczony przez członka współpracującego zestawu.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP

Po wywołaniu funkcji MQGET, która określa tę opcję, komunikat nie jest już uwzględniany przez żadne otwarte uchwyty w zestawie współpracujących uchwytów, które mają być oznaczone dla zestawu współpracującego. Komunikat jest nadal uważany za oznaczony na poziomie uchwytu, jeśli został oznaczony na poziomie uchwytu przed wywołaniem tego wywołania.

Użycie opcji MQGMO_UNMARK_BROWSE_CO_OP jest poprawne tylko z uchwytem zwróconego przez pomyślnie wywołanie programu MQOPEN z opcją MQOO_CO_OP. MQGET zakończy się powodzeniem, nawet jeśli komunikat nie został uznany za oznaczony przez współpracujący zestaw uchwytów.

Produkt MQGMO_UNMARK_BROWSE_CO_OP nie jest poprawny w przypadku wywołania MQGET bez przeglądania lub z dowolną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

MQGMO_UNMARK_BROWSE_HANDLE

Po wywołaniu programu MQGET, który określa tę opcję, znaleziony komunikat nie jest już traktowany jako oznaczony przez ten uchwyt.

Wywołanie powiedzie się nawet wtedy, gdy komunikat nie jest oznaczony dla tego uchwytu.

Ta opcja nie jest poprawna w przypadku wywołania MQGET bez przeglądania lub w przypadku dowolnej z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

Opcje blokady: Następujące opcje odnoszą się do blokowania komunikatów w kolejce:

MQGMO_LOCK

Zablokuj przejrany komunikat, aby komunikat stał się niewidoczny dla innego uchwytu otwartego dla kolejki. Tę opcję można określić tylko wtedy, gdy zostanie podana jedna z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Dla każdego uchwytu kolejki może być zablokowany tylko jeden komunikat. Komunikat może być komunikatem logicznym lub komunikatem fizycznym:

- Jeśli zostanie określona opcja MQGMO_COMPLETE_MSG, wszystkie segmenty komunikatów, które tworzą komunikat logiczny, zostaną zablokowane do uchwytu kolejki. Wszystkie komunikaty muszą być obecne w kolejce i muszą być dostępne do pobrania.
- Jeśli parametr MQGMO_COMPLETE_MSG zostanie pominięty, do uchwytu kolejki zostanie zablokowany tylko pojedynczy komunikat fizyczny. Jeśli ten komunikat stanie się segmentem komunikatu logicznego, zablokowany segment uniemożliwia innym aplikacjom korzystanie z programu MQGMO_COMPLETE_MSG w celu pobrania lub przeglądania komunikatu logicznego.

Zablokowana wiadomość jest zawsze tą, która znajduje się pod kursorem przeglądania. Komunikat może zostać usunięty z kolejki za pomocą późniejszego wywołania MQGET, które określa opcję MQGMO_MSG_UNDER_CURSOR. Inne wywołania programu MQGET korzystające z uchwytu kolejki mogą również usunąć komunikat (na przykład wywołanie określające identyfikator komunikatu zablokowanego).

Jeśli wywołanie zwróci kod zakończenia MQCC_FAILED, lub MQCC_WARNING z kodem przyczyny MQRC_TRUNCATED_MSG_FAILED, żaden komunikat nie jest zablokowany.

Jeśli aplikacja nie usunie komunikatu z kolejki, blokada zostanie zwolniona za pomocą jednego z następujących działań:

- Wywołaj kolejne wywołanie MQGET dla tego uchwytu, określając wartość MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT. Blokada zostanie zwolniona, jeśli wywołanie zakończy się z produktem MQCC_OK lub MQCC_WARNING. Jeśli wywołanie zakończy się MQCC_FAILED, komunikat pozostaje zablokowany. Istnieją jednak wyjątki:
 - Komunikat nie jest odblokowany, jeśli produkt MQCC_WARNING jest zwracany z produktem MQRC_TRUNCATED_MSG_FAILED.
 - Komunikat jest odblokowany, jeśli produkt MQCC_FAILED jest zwracany z produktem MQRC_NO_MSG_AVAILABLE.

Jeśli zostanie również określona wartość MQGMO_LOCK, zwrócony komunikat zostanie zablokowany. Jeśli parametr MQGMO_LOCK zostanie pominięty, po wywołaniu nie zostanie wyświetlony żaden zablokowany komunikat.

Jeśli zostanie podana wartość MQGMO_WAIT, a komunikat nie zostanie natychmiast wyświetlony, oryginalny komunikat zostanie odblokowany przed rozpoczęciem oczekiwania.

- Wydanie kolejnego wywołania MQGET dla tego uchwytu, z MQGMO_BROWSE_MSG_UNDER_CURSOR, bez MQGMO_LOCK. Blokada zostanie zwolniona, jeśli wywołanie zakończy się z produktem MQCC_OK lub MQCC_WARNING. Jeśli wywołanie zakończy się MQCC_FAILED, komunikat pozostaje zablokowany. Jednak zastosowanie ma następujący wyjątek:
 - Komunikat nie jest odblokowany, jeśli produkt MQCC_WARNING jest zwracany z produktem MQRC_TRUNCATED_MSG_FAILED.
- Wydanie kolejnego wywołania MQGET dla tego uchwytu z produktem MQGMO_UNLOCK.
- Wywołanie wywołania MQCLOSE za pomocą uchwytu. MQCLOSE może być niejawnym, spowodowany zakończeniem aplikacji.

Żadna specjalna opcja MQOPEN nie jest wymagana do określenia MQGMO_LOCK, innego niż MQOO_BROWSE, co jest wymagane do określenia towarzyszącej opcji przeglądania.

MQGMO_LOCK nie jest poprawna z żadną z następujących opcji:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Produkt MQGMO_LOCK nie jest możliwy, jeśli klient IBM MQ jest używany w systemie HP Integrity NonStop Server do menedżera kolejek produktu z/OS, gdy jest on koordynowany przez program TMF.

MQGMO_UNLOCK

Wiadomość, która ma zostać odblokowana, musi być wcześniej zablokowana przez wywołanie MQGET z opcją MQGMO_LOCK. Jeśli dla tego uchwytu nie jest zablokowany żaden komunikat, wywołanie zostanie zakończone z MQCC_WARNING i MQRC_NO_MSG_LOCKED.

Parametry *MsgDesc*, *BufferLength*, *Bufferi DataLength* nie są sprawdzane ani zmieniane, jeśli użytkownik poda MQGMO_UNLOCK. W programie *Buffer* nie jest zwracany żaden komunikat.

Nie jest wymagana żadna specjalna opcja otwierania w celu określenia MQGMO_UNLOCK (choć produkt MQ00_BROWSE jest wymagany do wydania żądania blokady na pierwszym miejscu).

Ta opcja nie jest poprawna z następującymi opcjami, z wyjątkiem następujących:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Obie te opcje są przyjmowane, niezależnie od tego, czy zostały określone, czy nie.

Opcje danych komunikatu: Następujące opcje odnoszą się do przetwarzania danych komunikatu, gdy komunikat jest odczytywany z kolejki:

MQGMO_ACCEPT_TRUNCATED_MSG

Jeśli bufor komunikatów jest zbyt mały, aby pomieścić pełny komunikat, należy zezwolić na wywołanie funkcji MQGET w celu wypełnienia buforu. MQGET zapełnia bufor tak, jak wiele komunikatów, jakie może on uzyskać. Generuje kod zakończenia ostrzeżenia i kończy przetwarzanie. Oznacza to, że:

- Podczas przeglądania komunikatów kursor przeglądania jest zaawansowany do zwróconego komunikatu.
- Podczas usuwania komunikatów zwrócony komunikat jest usuwany z kolejki.
- Kod przyczyny MQRC_TRUNCATED_MSG_ACCEPTED jest zwracany, jeśli nie wystąpi inny błąd.

Bez tej opcji bufor jest nadal wypełniany jako część komunikatu, który może być wstrzymany. Kod zakończenia ostrzeżenia został wygenerowany, ale przetwarzanie nie zostało zakończone. Oznacza to, że:

- Podczas przeglądania komunikatów kursor przeglądania nie jest zaawansowany.
- Podczas usuwania komunikatów komunikat nie jest usuwany z kolejki.
- Kod przyczyny MQRC_TRUNCATED_MSG_FAILED jest zwracany, jeśli nie wystąpi inny błąd.

MQGMO_CONVERT

Ta opcja przekształca dane aplikacji w komunikacie w taki sposób, aby były zgodne z wartościami *CodedCharSetId* i *Encoding* określonymi w parametrze *MsgDesc* w wywołaniu MQGET. Dane są przekształcane, zanim zostaną skopiowane do parametru *Buffer*.

Pole *Format* określone podczas umieszczania komunikatu jest przejęte przez proces konwersji w celu określenia rodzaju danych w komunikacie. Dane komunikatu są konwertowane przez menedżera kolejek dla formatów wbudowanych oraz przez wyjście pisane przez użytkownika dla innych formatów. Szczegółowe informacje na temat wyjścia konwersji danych znajdują się w sekcji ["Konwersja danych"](#) na stronie 919.

- Jeśli konwersja powiedzie się, pola *CodedCharSetId* i *Encoding* określone w parametrze *MsgDesc* pozostaną niezmiennicze w wyniku wywołania funkcji MQGET.

- Jeśli tylko konwersja nie powiedzie się, zwracane są dane komunikatu bez konwersji pól *CodedCharSetId* i *Encoding* w programie *MsgDesc* są ustawiane na wartości dla nieprzekształconego komunikatu. W tym przypadku kod zakończenia to MQCC_WARNING .

W obu przypadkach w tych polach opisano identyfikator zestawu znaków i kodowanie danych komunikatu zwracanych w parametrze *Buffer* .

See the *Format* field described in “MQMD-deskryptor komunikatu” na stronie 417 for a list of format names for which the queue manager performs the conversion.

Opcje grupy i segmentu: Następujące opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Przed opisami opcji, oto kilka definicji ważnych terminów:

Komunikat fizyczny

Komunikat fizyczny to najmniejsza jednostka informacji, która może zostać umieszczona w kolejce lub usunięta z kolejki. Często odpowiada on informacjom określonym lub pobranym na pojedynczym wywołaniu MQPUT, MQPUT1 lub MQGET . Każdy komunikat fizyczny ma własny deskryptor komunikatu MQMD. Zwykle komunikaty fizyczne wyróżniają się różniącymi się wartościami identyfikatora komunikatu, pola *MsgId* w MQMD. Menedżer kolejek nie wymusza innych wartości.

Komunikat logiczny

Komunikat logiczny jest pojedynczą jednostką informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny jest taki sam, jak komunikat fizyczny. Jeśli komunikaty logiczne są duże, ograniczenia systemowe mogą być zalecane lub konieczne, aby podzielić komunikat logiczny na dwa lub więcej komunikatów fizycznych nazywanych segmentami.

Komunikat logiczny, który został posegmentowany, składa się z dwóch lub większej liczby komunikatów fizycznych, które mają ten sam identyfikator grupy bez wartości NULL, pole *GroupId* w tabeli MQMD. Mają one ten sam numer kolejny komunikatu, *MsgSeqNumber* w polu MQMD. Segmenty są rozróżniane przez różne wartości dla przesunięcia segmentu, *Offset* w polu MQMD. Przesunięcie segmentu to przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dopuszczona przez aplikację wysyłającą, ma również identyfikator grupy o niezerowej wartości NULL. W tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zahamowana przez aplikację wysyłającą, mają identyfikator grupy o wartości NULL, MQGI_NONE, chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy

Grupa komunikatów jest zestawem jednego lub większej liczby komunikatów logicznych, które mają ten sam niepusty identyfikator grupy. Komunikaty logiczne w grupie są rozróżniane przez różne wartości dla numeru kolejnego komunikatu. Numer kolejny jest liczbą całkowitą z zakresu od 1 do n, gdzie n jest liczbą komunikatów logicznych w grupie. Jeśli co najmniej jeden komunikat logiczny jest segmentowany, w grupie jest więcej niż n komunikatów fizycznych.

MQGMO_LOGICAL_ORDER

Produkt MQGMO_LOGICAL_ORDER kontroluje kolejność, w jakiej komunikaty są zwracane przez kolejne wywołania programu MQGET dla uchwytu kolejki. Opcja musi być podana w każdym wywołaniu.

Jeśli dla kolejnych wywołań programu MQGET określono wartość MQGMO_LOGICAL_ORDER dla tego samego uchwytu kolejki, komunikaty w grupach są zwracane w kolejności ich numerów kolejnych komunikatów. Segmenty komunikatów logicznych są zwracane w kolejności określonej przez ich przesunięcia segmentów. Kolejność ta może różnić się od kolejności, w jakiej komunikaty i segmenty występują w kolejce.

Uwaga: Określenie MQGMO_LOGICAL_ORDER nie ma żadnych negatywnych konsekwencji dla komunikatów, które nie należą do grup, a które nie są segmentami. W efekcie takie komunikaty są traktowane tak, jakby każdy należał do grupy komunikatów składającej się tylko z jednego komunikatu. Podczas pobierania komunikatów z kolejek zawierających mieszanie komunikatów

w grupach, segmentach komunikatów i nieposegmentowanych komunikatów, które nie są w grupach, można bezpiecznie określić wartość `MQGMO_LOGICAL_ORDER`.

W celu zwrócenia komunikatów w wymaganej kolejności menedżer kolejek zachowuje informacje o grupach i segmentach między kolejnymi wywołaniami produktu `MQGET`. Informacje o grupach i segmentach identyfikują bieżącą grupę komunikatów i bieżący komunikat logiczny dla uchwytu kolejki. Identyfikuje on także bieżącą pozycję w grupie i komunikacie logicznym oraz informację, czy komunikaty są pobierane w ramach jednostki pracy. Ponieważ menedżer kolejek zachowuje te informacje, aplikacja nie musi ustawiać informacji o grupach i segmentach przed każdym wywołaniem produktu `MQGET`. W szczególności oznacza to, że aplikacja nie musi ustawiać pól `GroupId`, `MsgSeqNumber` i `Offset` w `MQMD`. Jednak aplikacja musi poprawnie ustawić opcję `MQGMO_SYNCPOINT` lub `MQGMO_NO_SYNCPOINT` dla każdego wywołania.

Po otwarciu kolejki nie ma bieżącej grupy komunikatów i nie ma bieżącego komunikatu logicznego. Grupa komunikatów staje się bieżącą grupą komunikatów, gdy komunikat, który ma flagę `MQMF_MSG_IN_GROUP`, jest zwracany przez wywołanie `MQGET`. Gdy program `MQGMO_LOGICAL_ORDER` jest określony w kolejnych wywołaniach, grupa ta pozostaje grupą bieżącą do momentu zwrócenia komunikatu o następującej treści:

- `MQMF_LAST_MSG_IN_GROUP` bez `MQMF_SEGMENT` (oznacza to, że ostatni komunikat logiczny w grupie nie jest segmentowany), lub
- `MQMF_LAST_MSG_IN_GROUP` z `MQMF_LAST_SEGMENT` (oznacza to, że zwrócony komunikat jest ostatnim segmentem ostatniego komunikatu logicznego w grupie).

Gdy taki komunikat jest zwracany, grupa komunikatów zostaje zakończona, a po pomyślnym zakończeniu wywołania `MQGET` nie ma już bieżącej grupy. W podobny sposób komunikat logiczny staje się bieżącym komunikatem logicznym, gdy komunikat, który ma flagę `MQMF_SEGMENT`, jest zwracany przez wywołanie `MQGET`. Komunikat logiczny zostaje zakończony, gdy zostanie zwrócony komunikat z flagą `MQMF_LAST_SEGMENT`.

Jeśli nie zostaną określone żadne kryteria wyboru, kolejne wywołania programu `MQGET` zwracają w poprawnej kolejności komunikaty dla pierwszej grupy komunikatów w kolejce. Następnie zwracane są komunikaty dla drugiej grupy komunikatów itd. aż do momentu, w którym nie będzie już dostępnych komunikatów. Możliwe jest wybranie wybranych grup komunikatów, określając jedną lub więcej spośród następujących opcji w polu *MatchOptions*:

- `MQMO_MATCH_MSG_ID`
- `MQMO_MATCH_CORREL_ID`
- `MQMO_MATCH_GROUP_ID`

Opcje te są jednak skuteczne tylko wtedy, gdy nie istnieje żadna bieżąca grupa komunikatów ani komunikat logiczny. Więcej informacji na ten temat zawiera sekcja *MatchOptions* opisana w sekcji [“MQGMO-Opcje Get-message” na stronie 367](#).

Tabela 46 na stronie 388 przedstawia wartości pól `MsgId`, `CorrelId`, `GroupId`, `MsgSeqNumber` i `Offset`, dla których menedżer kolejek szuka podczas próby znalezienia komunikatu, który ma zostać zwrócony w wywołaniu `MQGET`. Reguły mają zastosowanie zarówno do usuwania komunikatów z kolejki, jak i do przeglądania komunikatów w kolejce. W tabeli: Tak lub Nie:

LOG ORD

Wskazuje, czy opcja `MQGMO_LOGICAL_ORDER` jest określona w wywołaniu.

Cur grp

Wskazuje, czy bieżąca grupa komunikatów istnieje przed wywołaniem.

Cur log msg

Wskazuje, czy bieżący komunikat logiczny istnieje przed wywołaniem.

Pozostałe kolumny

Pokaż wartości, dla których szuka się menedżer kolejek. Poprzednie oznaczanie wartości zwróconej dla pola w poprzednim komunikacie dla uchwytu kolejki.

Tabela 46. Opcje MQGET odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych							
Opcje określone przez użytkownika	Status grupy i dziennika-komunikat przed wywołaniem		Wartości, dla których menedżer kolejek szuka				
LOG ORD	Cur grp	Cur log msg	MsgId	CorrelId	GroupId	MsgSeqNumber	Offset
Tak	Nie	Nie	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>	1	0
Tak	Nie	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + długość poprzedniego segmentu
Tak	Tak	Nie	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + długość poprzedniego segmentu
Nie	Albo	Albo	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>	Sterowane przez produkt <i>MatchOptions</i>

Jeśli w kolejce znajduje się wiele grup komunikatów, które mogą zostać zwrócone, grupy te są zwracane w kolejności określonej przez pozycję w kolejce pierwszego segmentu pierwszego komunikatu logicznego w każdej grupie. Oznacza to, że komunikaty fizyczne, które mają numery kolejne komunikatów 1, oraz przesunięcia 0, określają kolejność, w jakiej zwracane są odpowiednie grupy.

Opcja MQGMO_LOGICAL_ORDER wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy komunikat logiczny lub segment w grupie jest pobierany w jednostce pracy, wszystkie pozostałe komunikaty i segmenty w grupie muszą zostać pobrane w ramach jednostki pracy, o ile ten sam uchwyt kolejki jest używany. Nie muszą one jednak być pobierane w ramach tej samej jednostki pracy. Umożliwia to grupie komunikatów składającej się z wielu komunikatów fizycznych, które mają zostać podzielone na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy komunikat logiczny lub segment w grupie *nie* jest pobierany w jednostce pracy, a używany jest ten sam uchwyt kolejki, żaden z pozostałych komunikatów logicznych i segmentów w grupie nie może być pobrany w ramach jednostki pracy.

Jeśli te warunki nie są spełnione, wywołanie MQGET kończy się niepowodzeniem z kodem przyczyny MQRC_INCONSISTENT_UOW.

Jeśli określono wartość MQGMO_LOGICAL_ORDER, wartość MQGMO podana w wywołaniu MQGET nie może być mniejsza niż MQGMO_VERSION_2, a wartość MQMD nie może być mniejsza niż

MQMD_VERSION_2. Jeśli ten warunek nie jest spełniony, wywołanie nie powiedzie się z kodem przyczyny MQRC_WRONG_GMO_VERSION lub MQRC_WRONG_MD_VERSION, jeśli jest to właściwe.

Jeśli parametr MQGMO_LOGICAL_ORDER *nie* jest określony dla kolejnych wywołań MQGET dla uchwytu kolejki, zwracane są komunikaty bez względu na to, czy należą one do grup komunikatów, czy też są to segmenty komunikatów logicznych. Oznacza to, że komunikaty lub segmenty z określonej grupy lub komunikatu logicznego mogą zostać zwrócone poza kolejką lub połączone z komunikatami lub segmentami z innych grup lub z komunikatów logicznych albo z komunikatami, które nie znajdują się w grupach i nie są segmentami. W takiej sytuacji konkretne komunikaty zwracane przez kolejne wywołania programu MQGET są kontrolowane przez opcje MQMO_* określone w tych wywołaniach (szczegółowe informacje na temat tych opcji zawiera opis pola *MatchOptions* (opisany w sekcji “MQGMO-Opcje Get-message” na stronie 367)).

Jest to technika, której można użyć do zrestartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić pola *GroupId*, *MsgSeqNumber*, *Offset* i *MatchOptions* na odpowiednie wartości, a następnie wywołać wywołanie MQGET z zestawem MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT, ale *bez* określania MQGMO_LOGICAL_ORDER. Jeśli to wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupach i segmentach, a kolejne wywołania programu MQGET używające tego uchwytu kolejki mogą określać MQGMO_LOGICAL_ORDER jako normalne.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołania MQGET, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołania MQPUT. Dodatkowo menedżer kolejek zachowuje oddzielne informacje dla:

- MQGET wywołuje te wywołania, które usuwają komunikaty z kolejki.
- Program MQGET wywołuje przeglądanie komunikatów w kolejce.

Dla każdego uchwytu kolejki aplikacja może łączyć wywołania MQGET, które określają MQGMO_LOGICAL_ORDER z wywołaniami MQGET, które nie są obsługiwane. Należy jednak zwrócić uwagę na następujące kwestie:

- Jeśli parametr MQGMO_LOGICAL_ORDER zostanie pominięty, każde pomyślne wywołanie funkcji MQGET spowoduje, że menedżer kolejek ustanie informacje o grupach i segmentach na wartości odpowiadające zwróconej wiadomości; ta wartość zastępuje istniejące informacje o grupach i segmentach zachowane przez menedżer kolejek dla uchwytu kolejki. Modyfikowane są tylko informacje odpowiednie do działania wywołania (przeglądanie lub usuwanie).
- Jeśli parametr MQGMO_LOGICAL_ORDER zostanie pominięty, wywołanie nie powiedzie się, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może się powieść z kodem zakończenia MQCC_WARNING. Tabela 47 na stronie 389 przedstawia różne przypadki, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia nie jest MQCC_OK, kodem przyczyny jest jeden z następujących kodów przyczyny (w zależności od przypadku):
 - MQRC_INCOMPLETE_GROUP
 - MQRC_INCOMPLETE_MSG
 - MQRC_INCONSISTENT_UOW

Uwaga: Menedżer kolejek nie sprawdza informacji o grupach i segmentach podczas przeglądania kolejki lub podczas zamykania kolejki, która została otwarta do przeglądania, ale nie jest wprowadzana; w takich przypadkach kod zakończenia jest zawsze MQCC_OK (nie zakłada się innych błędów).

Tabela 47. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach		
Bieżące połączenie to	Poprzednia nazwa to MQGET z MQGMO_LOGICAL_ORDER	Poprzednie wywołanie było MQGET bez MQGMO_LOGICAL_ORDER
MQGET z MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED

Tabela 47. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach (kontynuacja)

Bieżące połączenie to	Poprzednia nazwa to MQGET z MQGMO_LOGICAL_ORDER	Poprzednie wywołanie było MQGET bez MQGMO_LOGICAL_ORDER
MQGET bez MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE z niezakończonym komunikatem grupowym lub logicznym	MQCC_WARNING	MQCC_OK

Aplikacje, które mają pobierać komunikaty i segmenty w kolejności logicznej, są zalecane w celu określenia MQGMO_LOGICAL_ORDER, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia aplikację z potrzeby zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tą informacją. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż te, które są udostępniane przez opcję MQGMO_LOGICAL_ORDER. Można to osiągnąć, jeśli nie zostanie podana ta opcja. Aplikacja musi wtedy upewnić się, że pola *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* i *Offset* w katalogu MQMD oraz opcje MQMO_* w *MatchOptions* w MQGMO są ustawione poprawnie, przed każdym wywołaniem MQGET.

Na przykład aplikacja, która chce *przekazywać* komunikaty fizyczne, które otrzymuje, bez względu na to, czy te komunikaty znajdują się w grupach lub segmentach komunikatów logicznych, musi *nie* określać MQGMO_LOGICAL_ORDER. W złożonej sieci z wieloma ścieżkami między wysyłającym i odbierającym menedżery kolejek komunikaty fizyczne mogą być odbierane poza kolejnością. Jeśli nie określono ani parametru MQGMO_LOGICAL_ORDER, ani odpowiedniego MQPMO_LOGICAL_ORDER w wywołaniu MQPUT, aplikacja przekazujący może pobierać i przekazywać każdy komunikat fizyczny zaraz po jego nadejściu, bez konieczności oczekiwania na nadejście kolejnego komunikatu w kolejności logicznej.

You can specify MQGMO_LOGICAL_ORDER with any of the other MQGMO_* options, and with various of the MQMO_* options in appropriate circumstances (see preceding section).

- W systemie z/OS ta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć typ indeksu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, który musi być odwzorowany na mapy kolejek, musi być na poziomie CFLEVEL (3) lub wyższym.
- W systemach AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

MQGMO_COMPLETE_MSG

Wywołanie MQGET może zwrócić tylko pełny komunikat logiczny. Jeśli komunikat logiczny jest segmentowany, menedżer kolejek ponownie zestawia segmenty i zwraca do aplikacji pełny komunikat logiczny. Fakt, że komunikat logiczny był podzielony na segmenty, nie jest widoczny dla aplikacji pobierających dane.

Uwaga: Jest to jedyna opcja, która powoduje, że menedżer kolejek ma ponownie składać segmenty komunikatów. Jeśli ta opcja nie zostanie określona, segmenty są zwracane indywidualnie do aplikacji, jeśli są obecne w kolejce (i spełniają inne kryteria wyboru określone w wywołaniu MQGET). Aplikacje, które nie chcą otrzymywać poszczególnych segmentów, muszą zawsze określać MQGMO_COMPLETE_MSG.

Aby można było użyć tej opcji, aplikacja musi udostępnić bufor, który jest wystarczająco duży, aby pomieścić pełny komunikat, lub określić opcję MQGMO_ACCEPT_TRUNCATED_MSG.

Jeśli kolejka zawiera segmentowane komunikaty z brakującą część segmentów (być może dlatego, że zostały opóźnione w sieci i nie dotarły jeszcze do niej), podanie wartości MQGMO_COMPLETE_MSG uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak te segmenty komunikatów nadal przyczyniają się do wartości atrybutu kolejki produktu

CurrentQDepth . Oznacza to, że nie mogą istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość *CurrentQDepth* jest większa od zera.

W przypadku komunikatów *trwałych* menedżer kolejek może dokonać ponownego złożenia segmentów tylko w ramach jednostki pracy:

- Jeśli wywołanie MQGET działa w ramach zdefiniowanej przez użytkownika jednostki pracy, ta jednostka pracy jest używana. Jeśli wywołanie nie powiedzie się podczas procesu ponownego składania, menedżer kolejek przywróci w kolejce wszystkie segmenty, które zostały usunięte podczas ponownego składania. Jednak niepowodzenie nie uniemożliwia pomyślnego wykonania jednostki pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika i nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy na czas trwania wywołania. Jeśli wywołanie zakończy się pomyślnie, menedżer kolejek zatwierdza jednostkę pracy automatycznie (aplikacja nie musi wykonywać tej czynności). Jeśli wywołanie nie powiedzie się, menedżer kolejek wytworzy kopię zapasową jednostki pracy.
- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, ale istnieje zdefiniowana przez użytkownika jednostka pracy, menedżer kolejek nie może się ponownie zestawiać. Jeśli komunikat nie wymaga ponownego składania, wywołanie może się jeszcze powieść. Jeśli jednak komunikat wymaga ponownego składania, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_UOW_NOT_AVAILABLE.

W przypadku komunikatów *nietrwałych* menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu przeprowadzenia ponownego montażu.

Każdy komunikat fizyczny, który jest segmentem, ma własny deskryptor komunikatu. W przypadku segmentów tworzących pojedynczy komunikat logiczny większość pól w deskrytorze komunikatu jest taka sama dla wszystkich segmentów w komunikacie logicznym; zwykle jest to tylko pola *MsgId*, *Offset* i *MsgFlags* , które różnią się między segmentami w komunikacie logicznym. Jeśli jednak segment zostanie umieszczony w kolejce niedostarczonych komunikatów w pośrednim menedżerze kolejek, procedura obsługi DLQ pobiera komunikat określający opcję MQGMO_CONVERT , co może spowodować zmianę zestawu znaków lub kodowania segmentu. Jeśli procedura obsługi DLQ pomyślnie wyśle segment na swój sposób, segment może mieć zestaw znaków lub kodowanie, które różnią się od innych segmentów w komunikacie logicznym, gdy dany segment dociera do docelowego menedżera kolejek.

Komunikat logiczny składający się z segmentów, w których pola *CodedCharSetId* i *Encoding* różnią się, nie mogą być ponownie składane przez menedżera kolejek w jeden komunikat logiczny. Zamiast tego menedżer kolejek jest ponownie montuje i zwraca pierwsze kilka kolejnych segmentów na początku komunikatu logicznego, które mają takie same identyfikatory i kodowania zestawu znaków, a wywołanie MQGET kończy się kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_INCONSISTENT_CCIDS lub MQRC_INCONSISTENT_ENCODINGS, odpowiednio. Dzieje się tak niezależnie od tego, czy określono wartość MQGMO_CONVERT . Aby pobrać pozostałe segmenty, aplikacja musi ponownie wywołać wywołanie MQGET bez opcji MQGMO_COMPLETE_MSG , pobierając segmenty jeden po jednym. Produkt MQGMO_LOGICAL_ORDER może być używany do pobierania pozostałych segmentów w kolejności.

Aplikacja, która umieszcza segmenty, może również ustawić inne pola w deskrytorze komunikatu na wartości, które różnią się między segmentami. Nie ma jednak żadnej korzyści, jeśli aplikacja odbierający korzysta z programu MQGMO_COMPLETE_MSG w celu pobrania komunikatu logicznego. Gdy menedżer kolejek przedstawia komunikat logiczny, zwraca on w deskrytorze komunikatu wartości z deskryptora komunikatu dla segmentu *pierwszy* . Jedyne wyjątek to pole *MsgFlags* , które wskazuje menedżer kolejek w celu wskazania, że zmontowany komunikat jest jedynym segmentem.

Jeśli dla komunikatu raportu określono wartość MQGMO_COMPLETE_MSG , menedżer kolejek wykonuje specjalne przetwarzanie. Menedżer kolejek sprawdza kolejkę w celu sprawdzenia, czy wszystkie komunikaty raportu tego typu odnoszące się do różnych segmentów w komunikacie logicznym znajdują się w kolejce. Jeśli są one dostępne, można je pobrać jako pojedynczy komunikat, podając MQGMO_COMPLETE_MSG. Aby to było możliwe, komunikaty raportu muszą być generowane przez menedżer kolejek lub agent MCA obsługujący segmentację, albo aplikacja źródłowa musi zażądać

co najmniej 100 bajtów danych komunikatu (to znaczy odpowiednie opcje MQRO_*_WITH_DATA lub MQRO_*_WITH_FULL_DATA muszą zostać określone). Jeśli dla segmentu jest mniejsza niż pełna ilość danych aplikacji, brakujące bajty są zastępowane wartościami pustymi w zwróconej komunikacie raportu.

Jeśli parametr MQGMO_COMPLETE_MSG jest określony w produkcie MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR, kursor przeglądania musi być umieszczony w komunikacie, którego pole *Offset* w zmiennej MQMD ma wartość 0. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

MQGMO_COMPLETE_MSG oznacza MQGMO_ALL_SEGMENTS_AVAILABLE, które nie muszą być określone.

Wartość MQGMO_COMPLETE_MSG można określić przy użyciu dowolnej z pozostałych opcji produktu MQGMO_* poza produktem MQGMO_SYNCPOINT_IF_PERSISTENT, a także z dowolną z opcji MQMO_* (poza opcją MQMO_MATCH_OFFSET).

- W systemie z/OS ta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć typ indeksu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, który musi być odwzorowany na mapę kolejek na poziomie CFLEVEL (3) lub wyższym.
- W systemach AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

MQGMO_ALL_MSGS_AVAILABLE

Komunikaty w grupie stają się dostępne do pobrania tylko w przypadku, gdy dostępne są *wszystkie* komunikaty w grupie. Jeśli kolejka zawiera grupy komunikatów z niektórymi brakami komunikatów (być może dlatego, że zostały opóźnione w sieci i jeszcze nie dotarły do nich), podanie wartości MQGMO_ALL_MSGS_AVAILABLE uniemożliwia pobranie komunikatów należących do niekompletnych grup. Jednak te komunikaty nadal przyczyniają się do wartości atrybutu kolejki produktu *CurrentQDepth*. Oznacza to, że nie mogą istnieć żadne możliwe do pobrania grupy komunikatów, nawet jeśli wartość *CurrentQDepth* jest większa od zera. Jeśli nie ma innych komunikatów, które są dostępne do pobrania, kod przyczyny MQRC_NO_MSG_AVAILABLE jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie produktu MQGMO_ALL_MSGS_AVAILABLE zależy od tego, czy określono również wartość MQGMO_LOGICAL_ORDER :

- Jeśli określono obie opcje, produkt MQGMO_ALL_MSGS_AVAILABLE ma działanie *tylko*, gdy nie ma żadnej bieżącej grupy lub komunikatu logicznego. Jeśli istnieje *jest* bieżąca grupa lub komunikat logiczny, MQGMO_ALL_MSGS_AVAILABLE jest ignorowany. Oznacza to, że program MQGMO_ALL_MSGS_AVAILABLE może pozostawać w trakcie przetwarzania komunikatów w kolejności logicznej.
- Jeśli wartość MQGMO_ALL_MSGS_AVAILABLE jest określona bez opcji MQGMO_LOGICAL_ORDER, wówczas opcja MQGMO_ALL_MSGS_AVAILABLE zawsze ma działanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego komunikatu w grupie, aby można było usunąć pozostałe komunikaty w grupie.

Pomyślne zakończenie wywołania MQGET z określeniem MQGMO_ALL_MSGS_AVAILABLE oznacza, że w momencie wywołania funkcji MQGET wszystkie komunikaty w grupie znajdowały się w kolejce. Należy jednak pamiętać, że inne aplikacje mogą nadal usuwać komunikaty z grupy (grupa nie jest zablokowana dla aplikacji, która pobiera pierwszy komunikat w grupie).

Jeśli ta opcja zostanie pominięta, komunikaty należące do grup mogą być pobierane nawet wtedy, gdy grupa jest niekompletna.

MQGMO_ALL_MSGS_AVAILABLE oznacza MQGMO_ALL_SEGMENTS_AVAILABLE, które nie muszą być określone.

MQGMO_ALL_MSGS_AVAILABLE można określić za pomocą dowolnej z pozostałych opcji MQGMO_* oraz z dowolną z opcji MQMO*.

- W systemie z/OS ta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć typ indeksu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, który musi być odwzorowany na mapę kolejek na poziomie CFLEVEL (3) lub wyższym.
- W systemach AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

MQGMO_ALL_SEGMENTS_AVAILABLE

Segmenty w komunikacie logicznym stają się dostępne do pobrania tylko w przypadku, gdy dostępne są *wszystkie* segmenty w komunikacie logicznym. Jeśli kolejka zawiera segmentowane komunikaty z brakującą częścią segmentów (być może dlatego, że zostały one opóźnione w sieci i nie zostały jeszcze odebrane), podanie wartości MQGMO_ALL_SEGMENTS_AVAILABLE uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak segmenty te nadal przyczyniają się do wartości atrybutu kolejki produktu *CurrentQDepth*. Oznacza to, że nie można pobrać komunikatów logicznych, nawet jeśli wartość *CurrentQDepth* jest większa od zera. Jeśli nie ma innych komunikatów, które są dostępne do pobrania, kod przyczyny MQRC_NO_MSG_AVAILABLE jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie produktu MQGMO_ALL_SEGMENTS_AVAILABLE zależy od tego, czy określono również wartość MQGMO_LOGICAL_ORDER :

- Jeśli określono obie opcje, produkt MQGMO_ALL_SEGMENTS_AVAILABLE ma działanie *tylko*, gdy nie ma bieżącego komunikatu logicznego. Jeśli *jest* bieżącym komunikatem logicznym, MQGMO_ALL_SEGMENTS_AVAILABLE jest ignorowany. Oznacza to, że program MQGMO_ALL_SEGMENTS_AVAILABLE może pozostawać w trakcie przetwarzania komunikatów w kolejności logicznej.
- Jeśli wartość MQGMO_ALL_SEGMENTS_AVAILABLE jest określona bez opcji MQGMO_LOGICAL_ORDER, wówczas opcja MQGMO_ALL_SEGMENTS_AVAILABLE zawsze ma działanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego segmentu w komunikacie logicznym, aby możliwe było usunięcie pozostałych segmentów w komunikacie logicznym.

Jeśli ta opcja nie zostanie podana, segmenty komunikatów mogą być pobierane nawet wtedy, gdy komunikat logiczny jest niekompletny.

Podczas gdy zarówno produkty MQGMO_COMPLETE_MSG, jak i MQGMO_ALL_SEGMENTS_AVAILABLE wymagają, aby wszystkie segmenty były dostępne, zanim można będzie pobrać dowolny z nich, to pierwsza z nich zwraca cały komunikat, podczas gdy te ostatnie umożliwiają pobranie segmentów po jednej stronie.

Jeśli dla komunikatu raportu określono wartość MQGMO_ALL_SEGMENTS_AVAILABLE, menedżer kolejek sprawdza kolejną w celu sprawdzenia, czy istnieje co najmniej jeden komunikat raportu dla każdego z segmentów, które składają się na pełny komunikat logiczny. Jeśli istnieje, warunek MQGMO_ALL_SEGMENTS_AVAILABLE jest spełniony. Jednak menedżer kolejek nie sprawdza typu *typ* komunikatów raportu, dlatego może istnieć mieszanka typów raportów w komunikatach raporty/dotyczących segmentów komunikatu logicznego. W rezultacie powodzenie produktu MQGMO_ALL_SEGMENTS_AVAILABLE nie oznacza, że MQGMO_COMPLETE_MSG zakończy się pomyślnie. Jeśli *jest* mieszaniną typów raportów przedstawianych dla segmentów konkretnego komunikatu logicznego, te komunikaty muszą zostać pobrane jeden po jednym.

Użytkownik może określić MQGMO_ALL_SEGMENTS_AVAILABLE z dowolną inną opcją MQGMO_*, a także z dowolną z opcji MQMO_*.

- W systemie z/OS ta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć typ indeksu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, który musi być odwzorowany na mapę kolejek na poziomie CFLEVEL (3) lub wyższym.
- W systemach AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

Opcje właściwości: Następujące opcje odnoszą się do właściwości komunikatu:

MQGMO_PROPERTIES_AS_Q_DEF

Właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskrypcorze komunikatu (lub rozszerzeniu), powinny być reprezentowane zgodnie z definicją atrybutu kolejki produktu *PropertyControl*. Jeśli *MsgHandle* jest dostępna, ta opcja jest ignorowana, a właściwości komunikatu są dostępne za pośrednictwem konsoli *MsgHandle*, chyba że wartością atrybutu kolejki *PropertyControl* jest MQPROP_FORCE_MQRFH2.

Jest to działanie domyślne w sytuacji, gdy nie są określone opcje właściwości.

MQGMO_PROPERTIES_IN_HANDLE

Właściwości komunikatu powinny być udostępniane za pośrednictwem konsoli *MsgHandle*. Jeśli nie udostępniono uchwytu komunikatu, wywołanie zakończy się niepowodzeniem z następującej przyczyny: MQRC_HMSG_ERROR.

Uwaga: Jeśli komunikat zostanie później odczytany przez aplikację, która nie utworzy uchwytu komunikatu, menedżer kolejek umieszcza wszystkie właściwości komunikatu w strukturze produktu MQRFH2. Może się okazać, że obecność nieoczekiwanego nagłówka MQRFH2 zakłóca zachowanie istniejącej aplikacji.

MQGMO_NO_PROPERTIES

Nie zostaną pobrane żadne właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskrypcorze komunikatu (lub rozszerzeniu). Jeśli zostanie podana wartość *MsgHandle*, zostanie ona zignorowana.

MQGMO_PROPERTIES_FORCE_MQRFH2

Właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskrypcorze komunikatu (lub rozszerzeniu), powinny być reprezentowane za pomocą nagłówków MQRFH2. Zapewnia to kompatybilność z wcześniejszymi wersjami aplikacji, które oczekują na pobranie właściwości, ale nie mogą zostać zmienione w celu użycia uchwytów komunikatów. Jeśli udostępniony zostanie element *MsgHandle*, zostanie on zignorowany.

MQGMO_PROPERTIES_COMPATIBILITY

Jeśli komunikat zawiera właściwość z przedrostkiem "mcd.", "jms.", "usr." lub "mqext.", wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2. W przeciwnym razie wszystkie właściwości komunikatu z wyjątkiem tych, które są zawarte w deskrypcorze komunikatu lub w rozszerzeniu, są usuwane i nie są już dostępne dla aplikacji.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

MQGMO_NONE

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Program MQGMO_NONE pomaga w dokumentacji programu; nie jest planowane używanie tej opcji razem z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową pola *Options* jest MQGMO_NO_WAIT plus MQGMO_PROPERTIES_AS_Q_DEF.

Reserved1 (MQCHAR)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Reserved2 (MQLONG)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż **MQGMO_VERSION_4**.

ResolvedQName (MQCHAR48)

Jest to pole wyjściowe, które menedżer kolejek ustawia na nazwę lokalną kolejki, z której został pobrany komunikat, zgodnie z definicją w lokalnym menedżerze kolejek. Wartość ta różni się od nazwy używanej do otwarcia kolejki, jeśli:

- Kolejka aliasowa została otwarta (w takim przypadku nazwa kolejki lokalnej, do której został zwrócony alias, jest zwracana), lub
- Otwarto kolejkę modelową (w takim przypadku zwracana jest nazwa dynamicznej kolejki lokalnej).

Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

ReturnedLength (MQLONG)

Jest to pole wyjściowe, które menedżer kolejek ustawia na długość w bajtach danych komunikatu zwróconych przez wywołanie MQGET w parametrze *Buffer*. Jeśli menedżer kolejek nie obsługuje tej możliwości, parametr *ReturnedLength* jest ustawiony na wartość MQRL_UNDEFINED.

Gdy komunikaty są przekształcane między kodowaniami lub zestawami znaków, dane komunikatu mogą czasami zmieniać wielkość. Po powrocie z wywołania MQGET:

- Jeśli parametr *ReturnedLength* ma wartość *nie* MQRL_UNDEFINED, liczba bajtów zwracanych danych komunikatu jest podawana przez produkt *ReturnedLength*.
- Jeśli parametr *ReturnedLength* ma wartość MQRL_UNDEFINED, liczba bajtów zwracanych danych komunikatu jest zwykle podawana przez mniejszą wartość *BufferLength* i *DataLength*, ale może być *mniejsza niż* wartość ta, jeśli wywołanie MQGET zakończy się z kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED. Jeśli tak się stanie, nieistotne bajty w parametrze *Buffer* są ustawione na wartości NULL.

Zdefiniowane są następujące wartości specjalne:

MQRL_NIEZDEFINIOWANY

Długość zwróconych danych nie została zdefiniowana.

W systemie z/OSwartością zwracaną dla pola *ReturnedLength* jest zawsze MQRL_UNDEFINED.

Wartością początkową tego pola jest MQRL_UNDEFINED. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_3.

Segmentacja (MQCHAR)

Jest to flaga wskazująca, czy dozwolona jest dalsza segmentacja dla pobranego komunikatu. Ma jedną z następujących wartości:

MQSEG_INHIBITED

Segmentacja nie jest dozwolona.

MQSEG_ALLOWED

Segmentacja jest dozwolona.

W systemie z/OSmenedżer kolejek zawsze ustawia to pole na wartość MQSEG_INHIBITED.

To jest pole wyjściowe. Wartością początkową tego pola jest MQSEG_INHIBITED (mqseg_inhibited). To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

SegmentStatus (MQCHAR)

Jest to flaga wskazująca, czy pobrany komunikat jest segmentem komunikatu logicznego. Ma jedną z następujących wartości:

MQSS_NOT_A_SEGMENT

Komunikat nie jest segmentem.

Segment MQSS_SEGMENT

Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

MQSS_LAST_SEGMENT,

Komunikat jest ostatnim segmentem komunikatu logicznego.

Jest to także wartość zwracana, jeśli komunikat logiczny składa się tylko z jednego segmentu.

W systemie z/OSmenedżer kolejek zawsze ustawia to pole na wartość MQSS_NOT_A_SEGMENT.

To jest pole wyjściowe. Początkowa wartość tego pola to MQSS_NOT_A_SEGMENT. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Signal1 (MQLONG)

Jest to pole wejściowe, które jest używane tylko w połączeniu z opcją MQGMO_SET_SIGNAL; identyfikuje on sygnał, który ma być dostarczony w momencie, gdy dostępny jest komunikat.

Uwaga: Typ danych i użycie tego pola są określane przez środowisko. Z tego powodu aplikacje, które mają być używane do portu między różnymi środowiskami, nie mogą używać sygnałów.

- W systemie z/OS to pole musi zawierać adres bloku kontrolnego zdarzeń (Event Control Block-ECB). Przed wydaniem wywołania MQGET, ECB musi być rozliczony przez aplikację. Przechowywanie danych zawierających ECB nie może być zwalniane do czasu zamknięcia kolejki. The ECB is posted by the queue manager with one of the signal completion codes described. Te kody zakończenia są ustawiane w bitach od 2 do 31 ECB, a obszar zdefiniowany w programie z/OS odwzorowuje makro IHAECB jako przeznaczone dla kodu zakończenia użytkownika.
- We wszystkich innych środowiskach jest to pole zastrzeżone; jego wartość nie jest znacząca.

Kody zakończenia sygnału są następujące:

MQEC_MSG_PRZYBYŁ

Do kolejki dotarł odpowiedni komunikat. Ten komunikat nie został zarezerwowany dla programu wywołującego; musi zostać wysłane drugie żądanie MQGET, ale inna aplikacja może pobrać komunikat przed wysłaniem drugiego żądania.

MQEC_WAIT_INTERVAL_EXPIRED

Podana wartość *WaitInterval* utraciła ważność bez odpowiedniego przybycia komunikatu.

MQEC_WAIT_ANULOWANE

Czas oczekiwania został anulowany z powodu nieokreślonej przyczyny (takiej jak zakończenie menedżera kolejek lub wyłączenie kolejki). Wprowadź ponownie żądanie, jeśli chcesz uzyskać dalszą diagnozę.

MQEC_Q_MGR QUIESCING,

Oczekiwanie zostało anulowane, ponieważ menedżer kolejek wszedł w stan wygaszania (MQGMO_FAIL_IF QUIESCING podano w wywołaniu MQGET).

MQEC_CONNECTION QUIESCING

Oczekiwanie zostało anulowane, ponieważ połączenie zostało wprowadzone w stan wygaszania (MQGMO_FAIL_IF QUIESCING zostało określone w wywołaniu MQGET).

Wartość początkowa tego pola jest określana przez środowisko:

- W systemie z/OS wartością początkową jest wskaźnik pusty.
- We wszystkich innych środowiskach wartością początkową jest 0.

Signal2 (MQLONG)

Jest to pole wejściowe, które jest używane tylko w połączeniu z opcją MQGMO_SET_SIGNAL. Jest to pole zastrzeżone; jego wartość nie jest znacząca.

Wartością początkową tego pola jest 0.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQGMO_STRUC_ID

Identyfikator struktury opcji get-message.

Dla języka programowania C jest również zdefiniowana stała zmienna MQGMO_STRUC_ID_ARRAY; ma taką samą wartość jak MQGMO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQGMO_STRUC_ID.

Wersja (MQLONG)

Wersja jest numerem wersji struktury.

Wartość musi być jedną z następujących wartości:

MQGMO_VERSION_1

Struktura opcji komunikatu get-message w wersji Version-1 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_2

Struktura opcji get-message w wersji Version-2 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_3

Struktura opcji komunikatu get-message w wersji Version-3 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_4

Struktura opcji komunikatu get-message w wersji Version-4 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MQGMO_CURRENT_VERSION

Bieżąca wersja struktury opcji get-message.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQGMO_VERSION_1.

WaitInterval (MQLONG)

Jest to przybliżony czas, wyrażony w milisekundach, przez który wywołanie MQGET oczekuje na nadejście odpowiedniego komunikatu (to znaczy komunikat spełniający kryteria wyboru określone w parametrze *MsgDesc* wywołania MQGET).

Ważne: Nie ma oczekiwania lub opóźnienia, jeśli odpowiedni komunikat jest dostępny natychmiast.

Więcej informacji na ten temat zawiera sekcja *MsgId* opisana w sekcji “MQMD-deskryptor komunikatu” na stronie 417 . Jeśli po tym czasie nie dotarł żaden odpowiedni komunikat, wywołanie zakończy się niepowodzeniem z kodem MQCC_FAILED i kodem przyczyny MQRC_NO_MSG_AVAILABLE.

W systemie z/OSczas, przez jaki wywołanie MQGET rzeczywiście czeka, wpływa na obciążenie systemu i planowanie pracy i może różnić się między wartością określoną dla *WaitInterval* i około 100 milisekund większą niż *WaitInterval*.

Produkt *WaitInterval* jest używany w połączeniu z opcją MQGMO_WAIT lub MQGMO_SET_SIGNAL. Jest ona ignorowana, jeśli żadna z tych wartości nie została określona. Jeśli zostanie podana jedna z tych wartości, wartość *WaitInterval* musi być większa lub równa zero lub następująca wartość specjalna:

MQWI_UNLIMITED

Nieograniczony przedział czasu oczekiwania.

Wartością początkową tego pola jest 0.

Wartości początkowe i deklaracje języków dla MQGMO

Tabela 48. Początkowe wartości pól w MQGMO dla MQGMO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQGMO_STRUC_ID	'GMO-'
<i>Version</i>	MQGMO_VERSION_1	1
<i>Options</i>	MQGMO_NO_WAIT	0

Tabela 48. Początkowe wartości pól w MQGMO dla MQGMO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>WaitInterval</i>	Brak	0
<i>Signal1</i>	Brak	Pusty wskaźnik na z/OS ; 0 inaczej
<i>Signal2</i>	Brak	0
<i>ResolvedQName</i>	Brak	Pusty łańcuch lub odstępy
<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_INHIBITED	'¬'
<i>Reserved1</i>	Brak	'¬'
<i>MsgToken</i>	MQMTOK_BRAK	Wartości null
<i>ReturnedLength</i>	MQRL_NIEZDEFINIOWANY	-1
<i>Reserved2</i>	Brak	'¬'
<i>MsgHandle</i>	MQHM_NONE	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraParametr MQGMO_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     Options;           /* Options that control the action of */
                                /* MQGET */
    MQLONG     WaitInterval;      /* Wait interval */
    MQLONG     Signal1;           /* Signal */
    MQLONG     Signal2;           /* Signal identifier */
    MQCHAR48    ResolvedQName;    /* Resolved name of destination queue */
    /* Ver:1 */
    MQLONG     MatchOptions;      /* Options controlling selection */
                                /* criteria used for MQGET */
    MQCHAR     GroupStatus;       /* Flag indicating whether message */
                                /* retrieved is in a group */
    MQCHAR     SegmentStatus;     /* Flag indicating whether message */
                                /* retrieved is a segment of a logical */
                                /* message */
    MQCHAR     Segmentation;      /* Flag indicating whether further */
                                /* segmentation is allowed for the */
                                /* message retrieved */
    MQCHAR     Reserved1;         /* Reserved */
    /* Ver:2 */
}
```

```

MQBYTE16  MsgToken;          /* Message token */
MQLONG    ReturnedLength;    /* Length of message data returned */
                                   (bytes) */

/* Ver:3 */
MQLONG    Reserved2;         /* Reserved */
MQHMSG    MsgHandle;         /* Message handle */
/* Ver:4 */
};

```

- W systemie z/OS pole *Signal1* jest zadeklarowane jako PMQLONG.

Deklaracja języka COBOL

```

** MQGMO structure
10 MQGMO.
**   Structure identifier
15 MQGMO-STRUCID          PIC X(4).
**   Structure version number
15 MQGMO-VERSION         PIC S9(9) BINARY.
**   Options that control the action of MQGET
15 MQGMO-OPTIONS         PIC S9(9) BINARY.
**   Wait interval
15 MQGMO-WAITINTERVAL    PIC S9(9) BINARY.
**   Signal
15 MQGMO-SIGNAL1         PIC S9(9) BINARY.
**   Signal identifier
15 MQGMO-SIGNAL2         PIC S9(9) BINARY.
**   Resolved name of destination queue
15 MQGMO-RESOLVEDQNAME   PIC X(48).
**   Options controlling selection criteria used for MQGET
15 MQGMO-MATCHOPTIONS    PIC S9(9) BINARY.
**   Flag indicating whether message retrieved is in a group
15 MQGMO-GROUPSTATUS     PIC X.
**   Flag indicating whether message retrieved is a segment of a
**   logical message
15 MQGMO-SEGMENTSTATUS   PIC X.
**   Flag indicating whether further segmentation is allowed for the
**   message retrieved
15 MQGMO-SEGMENTATION    PIC X.
**   Reserved
15 MQGMO-RESERVED1       PIC X.
**   Message token
15 MQGMO-MSGTOKEN        PIC X(16).
**   Length of message data returned (bytes)
15 MQGMO-RETURNEDLENGTH  PIC S9(9) BINARY.
**   Reserved
15 MQGMO-RESERVED2       PIC S9(9) BINARY.
**   Message handle
15 MQGMO-MSGHANDLE       PIC S9(18) BINARY.

```

- W systemie z/OS pole *Signal1* jest zadeklarowane jako POINTER.

Deklaracja PL/I

```

dcl
1 MQGMO based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 Options          fixed bin(31),    /* Options that control the action of
                                   MQGET */
3 WaitInterval     fixed bin(31),    /* Wait interval */
3 Signal1          fixed bin(31),    /* Signal */
3 Signal2          fixed bin(31),    /* Signal identifier */
3 ResolvedQName    char(48),         /* Resolved name of destination
                                   queue */
3 MatchOptions     fixed bin(31),    /* Options controlling selection
                                   criteria used for MQGET */
3 GroupStatus      char(1),          /* Flag indicating whether message
                                   retrieved is in a group */
3 SegmentStatus    char(1),          /* Flag indicating whether message
                                   retrieved is a segment of a logical
                                   message */
3 Segmentation     char(1),          /* Flag indicating whether further
                                   segmentation is allowed for the
                                   message retrieved */
3 Reserved1        char(1),          /* Reserved */

```

```

3 MsgToken      char(16),      /* Message token */
3 ReturnedLength fixed bin(31); /* Length of message data returned
                                (bytes) */
3 Reserved2     fixed bin(31); /* Reserved */
3 MsgHandle     fixed bin(63); /* Message handle */

```

- W systemie z/OS pole *Signal1* jest zadeklarowane jako pointer.

Deklaracja High Level Assembler

```

MQGMO          DSECT
MQGMO_STRUCID  DS    CL4      Structure identifier
MQGMO_VERSION  DS    F        Structure version number
MQGMO_OPTIONS  DS    F        Options that control the action of
*                               MQGET
MQGMO_WAITINTERVAL DS    F      Wait interval
MQGMO_SIGNAL1  DS    F        Signal
MQGMO_SIGNAL2  DS    F        Signal identifier
MQGMO_RESOLVEDQNAME DS    CL48  Resolved name of destination queue
MQGMO_MATCHOPTIONS DS    F      Options controlling selection criteria
*                               used for MQGET
MQGMO_GROUPSTATUS DS    CL1    Flag indicating whether message
*                               retrieved is in a group
MQGMO_SEGMENTSTATUS DS    CL1  Flag indicating whether message
*                               retrieved is a segment of a logical
*                               message
MQGMO_SEGMENTATION DS    CL1    Flag indicating whether further
*                               segmentation is allowed for the message
*                               retrieved
MQGMO_RESERVED1 DS    CL1      Reserved
MQGMO_MSGTOKEN  DS    XL16     Message token
MQGMO_RETURNEDLENGTH DS    F    Length of message data returned (bytes)
MQGMO_RESERVED2 DS    F        Reserved
MQGMO_MSGHANDLE DS    D        Message handle
MQGMO_LENGTH    EQU    *-MQGMO
MQGMO_AREA      ORG    MQGMO
                DS     CL(MQGMO_LENGTH)

```

Wizualna deklaracja podstawowa

```

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-nagłówek informacji IMS

W poniższej tabeli podsumowano pola w strukturze.

Tabela 49. Pola w tabeli MQIIH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId

Tabela 49. Pola w tabeli MQIIH (kontynuacja)		
Pole	Opis	Temat
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQIIH	StrucLength
<i>Encoding</i>	Zarezerwowane	Kodowanie
<i>CodedCharSetId</i>	Zarezerwowane	CodedCharSetId
<i>Format</i>	Nazwa formatu MQ danych, które są następujące: MQIIH	Formatowanie
<i>Flags</i>	Flagi	Flagi
<i>LTermOverride</i>	Nadpisanie terminalu logicznego	LTermOverride
<i>MFSMapName</i>	Nazwa odwzorowania usług w formacie komunikatów	MFSMapName
<i>ReplyToFormat</i>	Nazwa formatu MQ komunikatu odpowiedzi	ReplyToFormat
<i>Authenticator</i>	RACF™ password or passticket	Authenticator
<i>TranInstanceId</i>	Identyfikator instancji transakcji	IdentyfikatorTranInstance
<i>TranState</i>	Stan transakcji	TranState
<i>CommitMode</i>	Tryb kontroli transakcji	CommitMode
<i>SecurityScope</i>	Zakres ochrony	SecurityScope
<i>Reserved</i>	Zarezerwowane	Zarezerwowane

Przegląd produktu MQIIH

Struktura MQIIH opisuje informacje nagłówka dla komunikatu wysłanego do programu IMS w ramach mostu IMS. W przypadku dowolnej platformy IBM MQ obsługiwanej przez użytkownika można utworzyć i przestać komunikat zawierający strukturę MQIIH, ale tylko menedżer kolejek produktu IBM MQ for z/OS może używać mostu IMS. Oznacza to, że aby komunikat mógł zostać wysłany do programu IMS z menedżera kolejek innego niż z/OS, sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek produktu z/OS, przez który komunikat może być kierowany.

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ.

Cel: Struktura MQIIH opisuje informacje, które muszą być obecne na początku komunikatu wysyłanego do mostu IMS za pomocą programu IBM MQ for z/OS.

Nazwa formatu: MQFMT_IMS.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQIIH i danych komunikatu aplikacji:

- Aplikacje, które łączą się z menedżerem kolejek, do którego należy kolejka mostu IMS, muszą udostępniać strukturę MQIIH, która znajduje się w zestawie znaków i kodowaniu menedżera kolejek. Wynika to z faktu, że konwersja danych struktury MQIIH nie jest wykonywana w tym przypadku.
- Aplikacje, które łączą się z innymi menedżerami kolejek, mogą udostępniać strukturę MQIIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań; odbierający agent kanału komunikatów połączony z menedżerem kolejek, który jest właścicielem kolejki mostu IMS, przekształca tabelę MQIIH.
- Dane komunikatu aplikacji zgodnie ze strukturą MQIIH muszą być w tym samym zestawie znaków i kodowaniu co struktura MQIIH. Nie należy używać pól *CodedCharSetId* i *Encoding* w strukturze MQIIH, aby określić zestaw znaków i kodowanie danych komunikatu aplikacji.

Należy podać wyjście konwersji danych, aby przekształcić dane komunikatu aplikacji, jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek.

Pola dla tabeli MQIIH

Struktura MQIIH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Element uwierzytelniający (MQCHAR8)

Jest to hasło RACF lub PassTicket. Jest ona opcjonalna. Jeśli zostanie podana, jest ona używana z identyfikatorem użytkownika w kontekście zabezpieczeń MQMD w celu zbudowania znacznika UTOKEN, który jest wysyłany do produktu IMS w celu udostępnienia kontekstu zabezpieczeń. Jeśli nie zostanie podany, identyfikator użytkownika zostanie użyty bez weryfikacji. Zależy to od ustawienia przełączników RACF, które mogą wymagać obecności elementu uwierzytelniającego.

Opcja ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL. Można użyć następującej wartości specjalnej:

MQIAUT_NONE

Brak uwierzytelniania.

W przypadku języka programowania C zdefiniowana jest również stała MQIAUT_NONE_ARRAY; ma ona taką samą wartość jak MQIAUT_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez parametr MQ_AUTHENTICATOR_LENGTH. Wartością początkową tego pola jest MQIAUT_NONE.

CodedCharSetId (MQLONG)

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

Identyfikator zestawu znaków dla obsługiwanych struktur, które są zgodne ze strukturą MQIIH, jest taki sam, jak struktura struktury MQIIH i jest pobierana z dowolnego poprzedzającego nagłówka MQ.

CommitMode (MQCHAR)

Jest to tryb zatwierdzania IMS. Więcej informacji na temat trybów zatwierdzania produktu IMS zawiera publikacja *OTMA Reference*. Wartość musi być jedną z następujących wartości:

MQICM_COMMIT_THEN_SEND

Zatwierdź następnie wyślij.

Ten tryb implikuje podwójne kolejkowanie danych wyjściowych, ale krótsze czasy zajętości regionu. Transakcje typu fast-path i conversational nie mogą być uruchamiane z tym trybem.

MQICM_SEND_THEN_COMMIT

Wyślij następnie zatwierdzenie.

Dowolna transakcja IMS zainicjowana w wyniku trybu kontroli transakcji MQICM_SEND_THEN_COMMIT działa w trybie RESPONSE bez względu na to, jak transakcja jest zdefiniowana w definicji systemu IMS (parametr MSGTYPE w makrze TRANSACT). Dotyczy to również transakcji zainicjowanych za pomocą przełącznika transakcyjnego.

Początkowa wartość tego pola to MQICM_COMMIT_THEN_SEND.

Kodowanie (MQLONG)

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

Kodowanie obsługiwanych struktur, które są zgodne ze strukturą MQIIH, jest takie samo jak struktura struktury MQIIH, która jest pobierana z dowolnego poprzedzającego nagłówka MQ.

Flagi (MQLONG)

Wartość flag musi być następująca:

MQIIH_NONE

Brak flag.

MQIIH_PASS_EXPIRATION

Komunikat odpowiedzi zawiera:

- Te same opcje raportu utraty ważności, jak w przypadku komunikatu żądania
- Pozostały czas utraty ważności z komunikatu żądania bez korekty dla czasu przetwarzania mostu

Jeśli ta wartość nie jest ustawiona, czas utraty ważności jest ustawiony na *nieograniczony*.

MQIIH_REPLY_FORMAT_NONE

Ustawia wartość parametru MQIIH.Format odpowiedzi na wartość MQFMT_NONE.

MQIIH_IGNORE_PURG

Ustawia indyktor TMAMIPRG w przedrostku OTMA, który żąda, aby OTMA ignorowali wywołania PURG w bloku PCB TP dla transakcji CM0.

MQIIH_CM0_REQUEST_RESPONSE

W przypadku transakcji w trybie kontroli transakcji 0 (CM0) ta flaga ustawia indyktor TMAMHRSP w przedrostku OTMA. Ustawienie tego indykatora wymaga, aby program OTMA/IMS wygenerował komunikat DFS2082 RESPONSE MODE TRANSACTION ZAKOŃCZONY BEZ ODPOWIEDZI, gdy oryginalny program użytkowy IMS nie odpowie na IOPCB ani przełącznik komunikatów na inną transakcję.

Wartością początkową tego pola jest MQIIH_NONE.

Format (MQCHAR8)

Określa nazwę formatu danych MQ dla danych, które są zgodne ze strukturą MQIIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

LTermOverride (MQCHAR8)

Nadpisanie terminalu logicznego, które znajduje się w polu PCB we/wy. Jest ona opcjonalna; jeśli nie jest określona, używana jest nazwa TPIPE. Wartość ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL.

Długość tego pola jest podana przez wartość MQ_LTERM_OVERRIDE_LENGTH. Początkowa wartość tego pola to 8 znaków odstępu.

MFSMapName (MQCHAR8)

Nazwa odwzorowania usług w formacie komunikatów, umieszczana w polu PCB we/wy. Jest ono opcjonalne. Na wejściu reprezentuje identyfikator MID, na wyjściu reprezentuje on MOD. Wartość ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL.

Długość tego pola jest podana przez wartość MQ_MFS_MAP_NAME_LENGTH. Początkowa wartość tego pola to 8 znaków odstępu.

Format ReplyTo(MQCHAR8)

Jest to nazwa formatu produktu MQ komunikatu odpowiedzi, który jest wysyłany w odpowiedzi na bieżący komunikat. Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Aby przekształcić dane w komunikacie odpowiedzi przy użyciu funkcji MQGMO_CONVERT, należy określić wartość MQIIH.replyToFormat= MQFMT_STRING lub MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING. Wyjaśnienie korzystania z tych pól zawiera sekcja [“Format \(MQCHAR8\)” na stronie 433](#).

Jeśli wartość domyślna (MQIIH.replyToFormat= MQFMT_NONE) jest używana w komunikacie żądania, a komunikat odpowiedzi jest pobierany za pomocą komendy MQGMO_CONVERT, nie jest wykonywana konwersja danych.

Zarezerwowane (MQCHAR)

Jest to pole zastrzeżone. Musi być puste.

SecurityScope (MQCHAR)

Oznacza to, że wymagane jest przetwarzanie zabezpieczeń produktu IMS . Zdefiniowane są następujące wartości:

SPRAWDZANIE MQISS_CHECK

Sprawdź zasięg zabezpieczeń: ACEE jest budowany w regionie sterującym, ale nie w regionie zależnym.

MQISS_FULL

Pełny zakres ochrony: buforowany ACEE jest budowany w regionie sterowania, a niebuforowany ACEE jest budowany w regionie zależnym. Jeśli używana jest wartość MQISS_FULL, należy upewnić się, że identyfikator użytkownika, dla którego zbudowano ACEE, ma dostęp do zasobów używanych w regionie zależnym.

Jeśli dla tego pola nie określono ani MQISS_CHECK, ani MQISS_FULL, przyjmowana jest wartość MQISS_CHECK.

Wartością początkową tego pola jest MQISS_CHECK.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQIIH_STRUC_ID

Identyfikator struktury nagłówka informacyjnego produktu IMS .

Dla języka programowania C jest również zdefiniowana stała MQIIH_STRUC_ID_ARRAY. Ma ona taką samą wartość co identyfikator MQIIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQIIH_STRUC_ID.

StrucLength (MQLONG)

Jest to długość struktury MQIIH. Wartość musi być następująca:

MQIIH_LENGTH_1

Długość struktury nagłówka informacji IMS .

Początkowa wartość tego pola to MQIIH_LENGTH_1.

Identyfikator TranInstance(MQBYTE16)

Jest to identyfikator instancji transakcji. To pole jest używane przez komunikaty wyjściowe z produktu IMS, dlatego jest ignorowane przy pierwszym wejściu. Jeśli parametr *TranState* zostanie ustawiony na wartość MQITS_IN_CONVERSATION, to musi on być podany w następnych danych wejściowych i wszystkie kolejne wejścia, aby umożliwić IMS korelowanie komunikatów w poprawnej konwersacji. Można użyć następującej wartości specjalnej:

MQITII_NONE

Brak identyfikatora instancji transakcji.

W przypadku języka programowania C zdefiniowana jest również stała MQITII_NONE_ARRAY; ma ona taką samą wartość jak MQITII_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez wartość MQ_TRAN_INSTANCE_ID_LENGTH. Wartością początkową tego pola jest MQITII_NONE.

TranState (MQCHAR)

Wskazuje stan konwersacji IMS . Ta opcja jest ignorowana przy pierwszym wejściu, ponieważ żadna konwersacja nie istnieje. W kolejnych danych wejściowych wskazuje, czy konwersacja jest aktywna, czy nie. W przypadku wyjścia jest on ustawiany przez produkt IMS. Wartość musi być jedną z następujących wartości:

MQITS_IN_CONVERSATION,

W rozmowie.

MQITS_NOT_IN_CONVERSATION

Nie w rozmowie.

MQITS_ARCHITECTED

Zwróć dane stanu transakcji w postaci architected.

Ta wartość jest używana tylko z komendą IMS /DISPLAY TRAN . Zwraca on dane stanu transakcji

w postaci architected IMS zamiast postaci znaku.  Więcej informacji na ten temat zawiera sekcja [Zapisywanie programów transakcyjnych IMS za pomocą programu IBM MQ](#).

Wartością początkową tego pola jest MQITS_NOT_IN_CONVERSATION.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQIIH_VERSION_1

Numer wersji dla struktury nagłówka informacji IMS .

Następująca stała określa numer wersji bieżącej wersji:

MQIIH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka informacyjnego produktu IMS .

Początkowa wartość tego pola to MQIIH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQIIH

Tabela 50. Początkowe wartości pól w MQIIH dla MQIIH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQIIH_STRUC_ID	'IIH?'
<i>Version</i>	MQIIH_VERSION_1	1
<i>StrucLength</i>	MQIIH_LENGTH_1	84
<i>Encoding</i>	Brak	0
<i>CodedCharSetId</i>	Brak	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Flags</i>	MQIIH_NONE	0
<i>LTermOverride</i>	Brak	Puste
<i>MFSMapName</i>	Brak	Puste
<i>ReplyToFormat</i>	MQFMT_NONE	Puste
<i>Authenticator</i>	MQIAUT_NONE	Puste
<i>TranInstanceId</i>	MQITII_NONE	Wartości null
<i>TranState</i>	MQITS_NOT_IN_CONVERSATION	'?'
<i>CommitMode</i>	MQICM_COMMIT_THEN_SEND	'0'
<i>SecurityScope</i>	SPRAWDZANIE MQISS_CHECK	'C'
<i>Reserved</i>	Brak	'?'

Tabela 50. Początkowe wartości pól w MQIIH dla MQIIH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: 1. Symbol? reprezentuje pojedynczy pusty znak. 2. W języku programowania C: zmienna makraParametr MQIIH_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQIIH MyIIH = {MQIIH_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQIIH structure */
    MQLONG     Encoding;         /* Reserved */
    MQLONG     CodedCharSetId;   /* Reserved */
    MQCHAR8    Format;           /* MQ format name of data that follows
                                MQIIH */
    MQLONG     Flags;            /* Flags */
    MQCHAR8    LTermOverride;    /* Logical terminal override */
    MQCHAR8    MFSMapName;       /* Message format services map name */
    MQCHAR8    ReplyToFormat;    /* MQ format name of reply message */
    MQCHAR8    Authenticator;    /* RACF password or passticket */
    MQBYTE16   TranInstanceId;   /* Transaction instance identifier */
    MQCHAR     TranState;        /* Transaction state */
    MQCHAR     CommitMode;       /* Commit mode */
    MQCHAR     SecurityScope;    /* Security scope */
    MQCHAR     Reserved;        /* Reserved */
};
```

Deklaracja języka COBOL

```
** MQIIH structure
10 MQIIH.
** Structure identifier
15 MQIIH-STRUCID PIC X(4).
** Structure version number
15 MQIIH-VERSION PIC S9(9) BINARY.
** Length of MQIIH structure
15 MQIIH-STRUCLNGTH PIC S9(9) BINARY.
** Reserved
15 MQIIH-ENCODING PIC S9(9) BINARY.
** Reserved
15 MQIIH-CODEDCHARSETID PIC S9(9) BINARY.
** MQ format name of data that follows MQIIH
15 MQIIH-FORMAT PIC X(8).
** Flags
15 MQIIH-FLAGS PIC S9(9) BINARY.
** Logical terminal override
15 MQIIH-LTERMOVERRIDE PIC X(8).
** Message format services map name
15 MQIIH-MFSMAPNAME PIC X(8).
** MQ format name of reply message
15 MQIIH-REPLYTOFORMAT PIC X(8).
** RACF password or passticket
15 MQIIH-AUTHENTICATOR PIC X(8).
** Transaction instance identifier
15 MQIIH-TRANINSTANCEID PIC X(16).
** Transaction state
15 MQIIH-TRANSTATE PIC X.
** Commit mode
15 MQIIH-COMMITMODE PIC X.
** Security scope
15 MQIIH-SECURITYSCOPE PIC X.
```

```

**      Reserved
15 MQIIH-RESERVED      PIC X.

```

Deklaracja PL/I

```

dcl
1 MQIIH based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 StrucLength   fixed bin(31), /* Length of MQIIH structure */
3 Encoding      fixed bin(31), /* Reserved */
3 CodedCharSetId fixed bin(31), /* Reserved */
3 Format        char(8),      /* MQ format name of data that follows
                             MQIIH */
3 Flags        fixed bin(31), /* Flags */
3 LTermOverride char(8),      /* Logical terminal override */
3 MFSMapName    char(8),      /* Message format services map name */
3 ReplyToFormat char(8),      /* MQ format name of reply message */
3 Authenticator char(8),      /* RACF password or passticket */
3 TranInstanceId char(16),    /* Transaction instance identifier */
3 TranState     char(1),      /* Transaction state */
3 CommitMode    char(1),      /* Commit mode */
3 SecurityScope char(1),      /* Security scope */
3 Reserved      char(1);      /* Reserved */

```

Deklaracja High Level Assembler

```

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_CODEDCHARSETID DS    F  Reserved
MQIIH_FORMAT   DS    CL8      MQ format name of data that follows
*              MQIIH
MQIIH_FLAGS    DS    F        Flags
MQIIH_LTERM_OVERRIDE 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)

```

Wizualna deklaracja podstawowa

```

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-Zapytanie o opcje właściwości komunikatu

W poniższej tabeli podsumowano pola w strukturze. Struktura MQIMPO produktu -zapytanie o opcje właściwości komunikatu

Tabela 51. Pola w MQIMPO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem komendy MQINQMP	Opcje
<i>RequestedEncoding</i>	Kodowanie, do którego ma zostać przekształcona zapytana właściwość	RequestedEncoding
<i>RequestedCCSID</i>	Zestaw znaków właściwości inquired	RequestedCCSID
<i>ReturnedEncoding</i>	Kodowanie zwracanej wartości	ReturnedEncoding
<i>ReturnedCCSID</i>	Zestaw znaków zwracanej wartości	ReturnedCCSID
<i>Reserved1</i>	Zarezerwowane pole	ReturnedCCSID
<i>ReturnedName</i>	Nazwa zapytanej właściwości	ReturnedName
<i>TypeString</i>	Reprezentacja łańcuchowa typu danych właściwości	TypeString

Przegląd produktu MQIMPO

Struktura opcji sprawdzania właściwości komunikatu.

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ .

Przeznaczenie: Struktura MQIMPO umożliwia aplikacjom określanie opcji, które sterują sposobem uzyskiwania informacji o właściwościach komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQINQMP.

Zestaw znaków i kodowanie: Dane w tabeli MQIMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla MQIMPO

Sprawdź strukturę opcji właściwości komunikatu-pola

Struktura MQIMPO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

Zapytanie o strukturę opcji właściwości komunikatu-pole Opcje

Następujące opcje sterują działaniem komendy MQINQMP. Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Kombinacje opcji, które nie są poprawne, są oznaczane; wszystkie pozostałe kombinacje są poprawne.

Opcje danych wartości: Następujące opcje odnoszą się do przetwarzania danych wartości, gdy właściwość jest pobierana z komunikatu.

WARTOŚĆ MQIMPO_CONVERT_VALUE

Ta opcja żąda, aby wartość właściwości została przekształcona w taki sposób, aby była zgodna z wartościami *RequestedCCSID* i *RequestedEncoding* określonymi przed wywołaniem wywołania MQINQMP, zwracając wartość właściwości w obszarze *Value*.

- Jeśli konwersja powiedzie się, pola *ReturnedCCSID* i *ReturnedEncoding* są ustawione na takie same, jak *RequestedCCSID* i *RequestedEncoding* po powrocie z wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wartość właściwości zostanie zwrócona bez konwersji.

Jeśli właściwość jest łańcuchem, pola *ReturnedCCSID* i *ReturnedEncoding* są ustawiane na zestaw znaków i kodowanie nieprzekształconego łańcucha.

W tym przypadku kod zakończenia ma wartość MQCC_WARNING, a kod przyczyny MQRC_PROP_VALUE_NOT_CONVERTED. Cursor właściwości jest zaawansowany do zwróconej właściwości.

Jeśli wartość właściwości zostanie rozwinięta podczas konwersji, i przekracza wielkość parametru *Value*, zwracana jest wartość nieprzekształcona, a kod zakończenia MQCC_FAILED; kod przyczyny jest ustawiany na wartość MQRC_PROPERTY_VALUE_TOO_BIG.

Parametr *DataLength* wywołania MQINQMP zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmienny.

Ta opcja wymaga również, aby:

- Jeśli nazwa właściwości zawiera znak wieloznaczny, oraz
- Pole *ReturnedName* jest inicjowane za pomocą adresu lub przesunięcia dla zwróconej nazwy,

Zwracana nazwa jest przekształcana w taki sposób, aby była zgodna z wartościami *RequestedCCSID* i *RequestedEncoding*.

- Jeśli konwersja się powiedzie, pole *VSCCSID* produktu *ReturnedName* i kodowanie zwróconej nazwy są ustawiane na wartość wejściową *RequestedCCSID* i *RequestedEncoding*.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu lub ostrzeżenia, zwracana nazwa jest nieprzekształcona. W tym przypadku kod zakończenia ma wartość MQCC_WARNING, a kod przyczyny MQRC_PROP_NAME_NOT_CONVERTED.

Cursor właściwości jest zaawansowany do zwróconej właściwości. Wartość MQRC_PROP_VALUE_NOT_CONVERTED jest zwracana, jeśli nie została przekształcona wartość i nazwa.

Jeśli zwrócona nazwa zostanie rozwinięta podczas konwersji i zostanie przekroczona wielkość pola *VSBufsize* w *RequestedName*, zwrócony łańcuch zostanie przerobiony, a kod zakończenia MQCC_FAILED i kod przyczyny są ustawione na wartość MQRC_PROPERTY_NAME_TOO_BIG.

Pole *VSLength* struktury MQCHARV zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmienny.

TYP_KONTEKSTU MQIMPO_CONVERT_TYPE

Ta opcja żąda, aby wartość właściwości została przekształcona z jej bieżącego typu danych w typ danych określony w parametrze *Type* wywołania MQINQMP.

- Jeśli konwersja powiedzie się, parametr *Type* nie zostanie zmieniony podczas powrotu wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wywołanie zakończy się niepowodzeniem z powodu wartości MQRC_PROP_CONV_NOT_SUPPORTED. Cursor właściwości jest niezmienny.

Jeśli konwersja typu danych powoduje rozwinięcie wartości podczas konwersji, a wartość przekształcona przekracza wielkość parametru *Value*, zwracana jest wartość bez

konwersji, o kodzie zakończenia MQCC_FAILED, a kod przyczyny jest ustawiony na wartość MQRC_PROPERTY_VALUE_TOO_BIG.

Parametr *DataLength* wywołania MQINQMP zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmieniony.

Jeśli wartość parametru *Type* wywołania MQINQMP nie jest poprawna, wywołanie nie powiedzie się z powodu MQRC_PROPERTY_TYPE_ERROR.

Jeśli żądana konwersja typu danych nie jest obsługiwana, wywołanie kończy się niepowodzeniem z powodu wartości MQRC_PROP_CONV_NOT_SUPPORTED. Obsługiwane są następujące konwersje typów danych:

Typ danych właściwości	Obsługiwane docelowe typy danych
MQTYPE_BOOLEAN	MQTYPE_STRING, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_BYTE_STRING	MQTYPE_STRING
MQTYPE_INT8	MQTYPE_STRING, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT16	MQTYPE_STRING, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT32	MQTYPE_STRING, MQTYPE_INT64
MQTYPE_INT64	MQTYPE_STRING
MQTYPE_FLOAT32	MQTYPE_STRING MQTYPE_FLOAT64
MQTYPE_FLOAT64	MQTYPE_STRING
MQTYPE_STRING	MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64
MQTYPE_NULL	Brak

Ogólne reguły dotyczące obsługiwanych konwersji są następujące:

- Wartości właściwości liczbowych mogą być przekształcane z jednego typu danych na inny, pod warunkiem że w trakcie konwersji nie zostaną utracone żadne dane.

Na przykład wartość właściwości o typie danych MQTYPE_INT32 może być przekształcona w wartość o typie danych MQTYPE_INT64, ale nie może zostać przekształcona w wartość o typie danych MQTYPE_INT16.

- Wartość właściwości dowolnego typu danych może zostać przekształcona w łańcuch.
- Wartość właściwości łańcuchowej może zostać przekształcona w dowolny inny typ danych pod warunkiem, że łańcuch zostanie poprawnie sformatowany w celu konwersji. Jeśli aplikacja podejmie próbę przekształcenia wartości właściwości łańcuchowej, która nie jest poprawnie sformatowana, program IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR.
- Jeśli aplikacja podejmie próbę konwersji, która nie jest obsługiwana, program IBM MQ zwraca kod przyczyny MQRC_PROP_CONV_NOT_SUPPORTED.

Szczegółowe reguły przekształcania wartości właściwości z jednego typu danych na inny są następujące:

- Podczas konwersji wartości właściwości MQTYPE_BOOLEAN na łańcuch wartość TRUE jest przekształcana w łańcuch "TRUE", a wartość false jest przekształcana w łańcuch "FALSE".
- Podczas przekształcania wartości właściwości MQTYPE_BOOLEAN na liczbowy typ danych wartość TRUE jest przekształcana na wartość 1, a wartość FALSE jest przekształcana na zero.

- Podczas przekształcania wartości właściwości łańcuchowej w wartość MQTYPE_BOOLEAN łańcuch "TRUE" lub "1" jest przekształcany na TRUE, a łańcuch "FALSE" lub "0" jest przekształcany na wartość FALSE.

Należy zauważyć, że wielkość liter "TRUE" i "FALSE" nie jest rozróżniana.

Żaden inny łańcuch nie może zostać przekształcony; IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR.

- Podczas przekształcania wartości właściwości łańcuchowej na wartość z typem danych MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 lub MQTYPE_INT64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits
```

Znaczenia składników tego łańcucha są następujące:

blanks

Opcjonalne wiodące puste znaki

sign

Opcjonalny znak plus (+) lub znak minus (-).

digits

Ciągła sekwencja znaków cyfr (0-9). Musi istnieć co najmniej jeden znak cyfry.

Po sekwencji znaków cyfr łańcuch może zawierać inne znaki, które nie są znakami cyfr, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę całkowitą.

IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR, jeśli łańcuch nie jest poprawnie sformatowany.

- W przypadku przekształcania wartości właściwości łańcuchowej na wartość o typie danych MQTYPE_FLOAT32 lub MQTYPE_FLOAT64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

Znaczenia składników tego łańcucha są następujące:

blanks

Opcjonalne wiodące puste znaki

sign

Opcjonalny znak plus (+) lub znak minus (-).

digits

Ciągła sekwencja znaków cyfr (0-9). Musi istnieć co najmniej jeden znak cyfry.

e_char

Znak wykładnika, który jest albo "E", albo "e".

e_sign

Opcjonalny znak plus (+) lub znak minus (-) dla wykładnika.

e_digits

Ciągła sekwencja znaków cyfr (0-9) dla wykładnika. Jeśli łańcuch zawiera znak wykładnika, musi być obecny co najmniej jeden znak cyfry.

Po sekwencji znaków cyfr lub opcjonalnych znaków reprezentujących wykładnik, łańcuch może zawierać inne znaki, które nie są znakami cyfr, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje liczbę dziesiętną zmiennopozycyjną z wykładnikiem, który jest potęgą liczbą 10.

IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR, jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania wartości liczbowej właściwości w łańcuch wartość jest przekształcana na łańcuchową reprezentację wartości jako liczbę dziesiętną, a nie łańcuchową zawierającą znak ASCII dla tej wartości. Na przykład liczba całkowita 65 jest przekształcana w łańcuch "65", a nie łańcuch "A".
- Podczas przekształcania wartości właściwości łańcucha bajtowego w łańcuch każdy bajt jest przekształcany w dwa znaki szesnastkowe, które reprezentują bajt. Na przykład tablica bajtów {0xF1, 0x12, 0x00, 0xFF} jest przekształcana w łańcuch "F11200FF".

MQIMPO_QUERY_LENGTH

Zapytanie o typ i długość wartości właściwości. Długość jest zwracana w parametrze *DataLength* wywołania MQINQMP. Wartość właściwości nie jest zwracana.

Jeśli zostanie podany bufor *ReturnedName*, to pole *VSLength* struktury MQCHARV zostanie wypełnione nazwą właściwości. Nazwa właściwości nie jest zwracana.

Opcje iteracji: Następujące opcje są powiązane z iteracją nad właściwościami przy użyciu nazwy ze znakiem wieloznacznym

MQIMPO_INQ_FIRST

Sprawdź pierwszą właściwość, która jest zgodna z podaną nazwą. Po wywołaniu tej operacji na obiekcie, który jest zwracany, zostanie utworzony kursor.

Jest to wartość domyślna.

Opcja MQIMPO_INQ_PROP_UNDER_CURSOR może być następnie używana z wywołaniem MQINQMP, jeśli jest to wymagane, aby ponownie zapytać o tę samą właściwość.

Należy zauważyć, że istnieje tylko jeden kursor właściwości, dlatego jeśli nazwa właściwości określona w wywołaniu MQINQMP, zmienia kursor, zostanie zresetowana.

Ta opcja nie jest poprawna z jedną z następujących opcji:

MQIMPO_INQ_NEXT
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT

Sprawdź następną właściwość, która jest zgodna z podaną nazwą, kontynuując wyszukiwanie z kursora właściwości. Kursor jest awansowany do zwróconej właściwości.

Jeśli jest to pierwsze wywołanie MQINQMP dla podanej nazwy, zwracana jest pierwsza właściwość, która jest zgodna z podaną nazwą.

Opcja MQIMPO_INQ_PROP_UNDER_CURSOR może być następnie używana z wywołaniem MQINQMP, jeśli jest to wymagane, w celu ponownego uzyskania informacji na temat tej samej właściwości.

Jeśli właściwość pod kursorem została usunięta, komenda MQINQMP zwraca następną zgodną właściwość po usunięciu tej właściwości.

Jeśli właściwość zostanie dodana, która jest zgodna ze znakiem wieloznacznym, podczas gdy iteracja jest w toku, ta właściwość może lub nie może zostać zwrócona podczas kończenia iteracji. Ta właściwość jest zwracana po restarcie iteracji przy użyciu MQIMPO_INQ_FIRST.

Właściwość pasująca do znaku wieloznacznego, który został usunięty, podczas gdy iteracja była w toku, nie jest zwracana po jego usunięciu.

Ta opcja nie jest poprawna z jedną z następujących opcji:

MQIMPO_INQ_FIRST
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR

Pobieranie wartości właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana, przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany, gdy uchwyt komunikatu jest określony w polu *MsgHandle* obiektu MQGMO w wywołaniu MQGET, lub

gdy uchwyt komunikatu jest określony w polach *OriginalMsgHandle* lub *NewMsgHandle* struktury MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyną jest wartość MQRC_PROPERTY_NOT_AVAILABLE.

Ta opcja nie jest poprawna z jedną z następujących opcji:

MQIMPO_INQ_FIRST
MQIMPO_INQ_NEXT

Jeśli żadna z wcześniej opisanych opcji nie jest wymagana, można użyć następującej opcji:

MQIMPO_NONE

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

MQIMPO_NONE jest pomocna w dokumentacji programu; nie jest przeznaczona, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQIMPO_INQ_FIRST.

RequestedCCSID (MQLONG)

Sprawdź strukturę opcji właściwości komunikatu-pole RequestedCCSID

Zestaw znaków, w którym wartość właściwości inquired ma zostać przekształcona w łańcuch znaków, jeśli wartość ta jest łańcuchem znaków. Jest to również zestaw znaków, w którym ma zostać przekształcona *ReturnedName*, gdy określono wartość MQIMPO_CONVERT_VALUE lub MQIMPO_CONVERT_TYPE.

Wartością początkową tego pola jest MQCCSI_APPL.

RequestedEncoding (MQLONG)

Sprawdź strukturę opcji właściwości komunikatu-pole RequestedEncoding

Jest to kodowanie, w którym ma zostać przekształcona wartość właściwości inquired, gdy określono wartość MQIMPO_CONVERT_VALUE lub MQIMPO_CONVERT_TYPE.

Wartością początkową tego pola jest MQENC_NATIVE.

Reserved1 (MQCHAR)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem (pole 4 bajtowe).

ReturnedCCSID (MQLONG)

Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedCCSID

W przypadku danych wyjściowych jest to zestaw znaków wartości zwracanej, jeśli parametr *Type* wywołania MQINQMP ma wartość MQTYPE_STRING.

Jeśli określono opcję MQIMPO_CONVERT_VALUE, a konwersja się powiodła, pole *ReturnedCCSID* (w przypadku zwrotu) jest taką samą wartością, jak wartość przekazana.

Początkowa wartość tego pola wynosi zero.

ReturnedEncoding (MQLONG)

Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedEncoding

W przypadku danych wyjściowych jest to kodowanie zwracanej wartości.

Jeśli określono opcję MQIMPO_CONVERT_VALUE, a konwersja się powiodła, pole *ReturnedEncoding* (w przypadku zwrotu) jest taką samą wartością, jak wartość przekazana.

Wartością początkową tego pola jest MQENC_NATIVE.

ReturnedName (MQCHARV)

Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedName

Rzeczywista nazwa właściwości zapytania.

W przypadku wejścia bufor łańcuchowy może być przekazywany za pomocą pola *VSPtr* lub *VSOffset* struktury MQCHARV. Długość buforu łańcucha jest określona za pomocą pola *VSBufsize* struktury MQCHARV.

W przypadku powrotu z wywołania MQINQMP bufor łańcucha jest uzupełniany nazwą właściwości, która została zapytana, pod warunkiem, że bufor łańcuchowy był wystarczająco długi, aby mógł w pełni zawierać nazwę. Pole *VSLength* struktury MQCHARV jest wypełnione przez długość nazwy właściwości. Pole *VSCCSID* struktury MQCHARV jest wypełniane w celu wskazania zestawu znaków zwracanej nazwy, bez względu na to, czy konwersja nazwy nie powiodła się.

Jest to pole wejściowe/wyjściowe. Wartością początkową tego pola jest MQCHARV_DEFAULT.

StrucId (MQCHAR4)

Zapytanie o strukturę opcji właściwości komunikatu-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQIMPO_STRUC_ID,

Identyfikator zapytania o strukturę opcji właściwości komunikatu.

Dla języka programowania C jest również zdefiniowana stała MQIMPO_STRUC_ID_ARRAY. Ma ona taką samą wartość co identyfikator MQIMPO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQIMPO_STRUC_ID.

TypeString (MQCHAR8)

Sprawdź strukturę opcji właściwości komunikatu-pole TypeString

Reprezentacja łańcuchowa typu danych właściwości.

Jeśli właściwość została określona w nagłówku MQRFH2, a atrybut MQRFH2 dt nie został rozpoznany, to pole może zostać użyte do określenia typu danych właściwości. *TypeString* jest zwracany w kodowanym zestawie znaków 1208 (UTF-8) i jest to pierwsze osiem bajtów wartości atrybutu dt właściwości, które nie zostały rozpoznane.

To jest zawsze pole wyjściowe. Wartość początkowa tego pola jest łańcuchem pustym w języku programowania C, a 8 znaków odstępu w innych językach programowania.

Wersja (MQLONG)

Zapytanie o strukturę opcji właściwości komunikatu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQIMPO_VERSION_1

Numer wersji dla zapytania o strukturę opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQIMPO_CURRENT_VERSION

Bieżąca wersja struktury opcji sprawdzania właściwości komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQIMPO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQIMPO

Sprawdź strukturę opcji właściwości komunikatu-wartości początkowe

Tabela 52. Początkowe wartości pól w MQIMPO		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQIMPO_STRUC_ID,	'IMPO'
Version	MQIMPO_VERSION_1	1
Options	MQIMPO_INQ_FIRST	
RequestedEncoding	MQENC_NATIVE	
RequestedCCSID	MQCCSI_APPL	
ReturnedEncoding	MQENC_NATIVE	
ReturnedCCSID	0	
Reserved1	0	
ReturnedName	MQCHARV_DEFAULT	
TypeString	Pusty łańcuch lub odstęp	
Uwagi: 1. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. 2. W języku programowania C: zmienna makraParametr MQIMPO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQIMPO MyIMPO = {MQIMPO_DEFAULT};</pre>		

Deklaracja C

Zapytanie o strukturę opcji właściwości komunikatu-deklaracja języka C

```
typedef struct tagMQIMPO MQIMPO;
struct tagMQIMPO {
    MQCHAR4 StrucId;           /* Structure identifier */
    MQLONG  Version;           /* Structure version number */
    MQLONG  Options;           /* Options that control the action of
                               MQINQMP */
    MQLONG  RequestedEncoding; /* Requested encoding of Value */
    MQLONG  RequestedCCSID;    /* Requested character set identifier
                               of Value */
    MQLONG  ReturnedEncoding;  /* Returned encoding of Value */
    MQLONG  ReturnedCCSID;     /* Returned character set identifier
                               of Value */
    MQCHAR  Reserved1;         /* Reserved field */
    MQCHARV ReturnedName;      /* Returned property name */
    MQCHAR8 TypeString;        /* Property data type as a string */
};
```

Deklaracja języka COBOL

Sprawdź strukturę opcji właściwości komunikatu-deklaracja języka COBOL

```
** MQIMPO structure
10 MQIMPO.
** Structure identifier
15 MQIMPO-STRUCID          PIC X(4).
** Structure version number
15 MQIMPO-VERSION         PIC S9(9) BINARY.
```

```

** Options that control the action of MQINQMP
15 MQIMPO-OPTIONS PIC S9(9) BINARY.
** Requested encoding of VALUE
15 MQIMPO-REQUESTEDENCODING PIC S9(9) BINARY.
** Requested character set identifier of VALUE
15 MQIMPO-REQUESTEDCCSID PIC S9(9) BINARY.
** Returned encoding of VALUE
15 MQIMPO-RETURNEDENCODING PIC S9(9) BINARY.
** Returned character set identifier of VALUE
15 MQIMPO-RETURNEDCCSID PIC S9(9) BINARY.
** Reserved field
15 MQIMPO-RESERVED1
** Returned property name
15 MQIMPO-RETURNEDNAME.
** Address of variable length string
20 MQIMPO-RETURNEDNAME-VSPTR POINTER.
** Offset of variable length string
20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
** CCSID of variable length string
20 MQIMPO-RETURNEDNAME-VSCCSID PIC S9(9) BINARY.
** Property data type as string
15 MQIMPO-TYPESTRING PIC S9(9) BINARY.

```

Deklaracja PL/I

Zapytanie o strukturę opcji właściwości komunikatu-deklaracja języka PL/I

```

dcl
  1 MQIMPO based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 Options          fixed bin(31),    /* Options that control the
                                     action of MQINQMP */
    3 RequestedEncoding fixed bin(31),    /* Requested encoding of
                                     Value */
    3 RequestedCCSID    fixed bin(31),    /* Requested character set
                                     identifier of Value */
    3 ReturnedEncoding  fixed bin(31),    /* Returned encoding of
                                     Value */
    3 ReturnedCCSID     fixed bin(31),    /* Returned character set
                                     identifier of Value */
    3 Reserved1         fixed bin(31),    /* Reserved field */
    3 ReturnedName,     /* Returned property name */
      5 ReturnedName_VSPtr pointer,      /* Address of returned
                                     name */
      5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                     name */
      5 5 ReturnedName_VSCCSID fixed bin(31), /* CCSID of returned
                                     name */
    3 TypeString        char(8);          /* Property data type as
                                     string */

```

Deklaracja High Level Assembler

Zapytanie o strukturę opcji właściwości komunikatu-Składanie deklaracji języka

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-deskryptor komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 53. Pola w strukturze MQMD		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Report</i>	Opcje dla komunikatów raportu	Raport
<i>MsgType</i>	Typ komunikatu	MsgType
<i>Expiry</i>	Czas życia komunikatu	MQMD-pole Expiry (MQMD)
<i>Feedback</i>	Informacja zwrotna lub kod przyczyny	MQMD-pole opinii (Feedback)
<i>Encoding</i>	Kodowanie numeryczne danych komunikatu	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków danych komunikatu	CodedCharSetId
<i>Format</i>	Nazwa formatu danych komunikatu	Formatowanie
<i>Priority</i>	Priorytet komunikatu	Priorytet
<i>Persistence</i>	Trwałość komunikatu	Trwałość
<i>MsgId</i>	Identyfikator komunikatu	MQMD-pole MsgId
<i>CorrelId</i>	Identyfikator korelacji	CorrelId
<i>BackoutCount</i>	Liczba wycofanych	BackoutCount
<i>ReplyToQ</i>	Nazwa kolejki odpowiedzi	Kolejka_zwrotna
<i>ReplyToQMgr</i>	Nazwa menedżera kolejek odpowiedzi	ReplyToQMgr
<i>UserIdentifier</i>	Identyfikator użytkownika	UserIdentifier
<i>AccountingToken</i>	Token rozliczania	AccountingToken
<i>ApplIdentityData</i>	Dane aplikacji odnoszące się do tożsamości	Dane_tożsamości_aplikacji
<i>PutApplType</i>	Typ aplikacji, która wstawiła komunikat	Typ_aplikacji_wstawiającej
<i>PutApplName</i>	Nazwa aplikacji umieszczonej w komunikacie.	Nazwa_aplikacji_wstawiającej
<i>PutDate</i>	Data umieszczenia komunikatu	PutDate
<i>PutTime</i>	Czas umieszczenia komunikatu	PutTime
<i>ApplOriginData</i>	Dane dotyczące wniosku dotyczące pochodzenia	Dane_pochodzenia_aplikacji
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQMD_VERSION_2.		
<i>GroupId</i>	Identyfikator grupy	GroupId

Tabela 53. Pola w strukturze MQMD (kontynuacja)

Pole	Opis	Temat
<i>MsgSeqNumber</i>	Numer kolejny komunikatu logicznego w grupie	<u>Numer_kolejny_komunikatu</u>
<i>Offset</i>	Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego	<u>Depozycja</u>
<i>MsgFlags</i>	Flagi komunikatu	<u>MQMD-pole MsgFlags</u>
<i>OriginalLength</i>	Długość oryginalnego komunikatu	<u>OriginalLength</u>

Przegląd deskryptora MQMD

Dostępność: Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQMD zawiera informacje sterujące, które towarzyszą danych aplikacji podczas przesyłania komunikatów między aplikacjami wysyłającym i odbierającym. Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQGET, MQPUT i MQPUT1.

Wersja: Bieżąca wersja deskryptora MQMD to MQMD_VERSION_2. Aplikacje, które mają być przenośne między kilkoma środowiskami, muszą mieć pewność, że wymagana wersja deskryptora MQMD jest obsługiwana we wszystkich środowiskach, których to dotyczy. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję deskryptora MQMD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQMD_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *Version* na numer wersji wymaganej wersji.

Deklaracja dla struktury version-1 jest dostępna z nazwą MQMD1.

Zestaw znaków i kodowanie: Dane w strukturze MQMD muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one podawane przez atrybut menedżera kolejek produktu *CodedCharSetId* i atrybut MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako IBM MQ MQI client, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Jeśli menedżery kolejek wysyłających i odbierających korzystają z różnych zestawów znaków lub kodowań, dane w strukturze MQMD są przekształcane automatycznie. Nie jest konieczne, aby aplikacja przekształciła deskryptor MQMD.

Korzystanie z różnych wersji deskryptora MQMD: element MQMD w wersji version-2 jest równoważny z użyciem deskryptora MQMD z wersji version-1 i z prefixem danych komunikatu ze strukturą MQMDE. Jeśli jednak wszystkie pola w strukturze MQMDE mają swoje wartości domyślne, można pominąć MQMDE. version-1 MQMD plus MQMDE są używane w sposób opisany poniżej:

- W przypadku wywołań MQPUT i MQPUT1, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, aplikacja może opcjonalnie prefiksować dane komunikatu za pomocą MQMDE, ustawiając pole *Format* w strukturze MQMD na MQFMT_MD_EXTENSION, aby wskazać, że jest obecna tabela MQMDE. Jeśli aplikacja nie udostępnia wywołania MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w MQMDE.

Uwaga: Kilka pól istniejących w strukturze MQMD version-2, ale nie version-1 MQMD, są polami wejściowymi/wyjściowymi w wywołaniach MQPUT i MQPUT1. Jednak menedżer kolejek *nie* zwraca żadnych wartości w równoważnych polach w MQMDE na danych wyjściowych z wywołań MQPUT i MQPUT1. Jeśli aplikacja wymaga tych wartości wyjściowych, musi ona używać deskryptora MQMD z wersji version-2.

- W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, menedżer kolejek prefiksuje komunikat zwrócony przez produkt MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Pole *Format* w strukturze MQMD będzie miało wartość MQFMT_MD_EXTENSION, aby wskazać, że jest obecna MQMDE.

Wartości domyślne używane przez menedżera kolejek dla pól w tabeli MQMDE są takie same, jak początkowe wartości tych pól, które są wyświetlane w programie [Tabela 58 na stronie 475](#).

Jeśli komunikat znajduje się w kolejce transmisji, niektóre pola w strukturze MQMD są ustawiane na określone wartości. Szczegółowe informacje zawiera sekcja [“MQXQH-nagłówki kolejki transmisji” na stronie 623](#).

Kontekst komunikatu: Niektóre pola w strukturze MQMD zawierają kontekst komunikatu. Istnieją dwa typy kontekstu komunikatu: *kontekst tożsamości* i *kontekst źródłowy*. Zwykle:

- Kontekst tożsamości odnosi się do aplikacji, która *pierwotnie* umieszczała komunikat
- Kontekst źródłowy odnosi się do aplikacji, która *ostatnio* umieszczała komunikat.

Te dwie aplikacje mogą być tą samą aplikacją, ale mogą to być także różne aplikacje (na przykład, gdy komunikat jest przekazywany z jednej aplikacji do innej).

Chociaż kontekst tożsamości i pochodzenia zwykle ma opisane znaczenie, treść obu typów pól kontekstu w strukturze MQMD zależy od opcji MQPMO_*_CONTEXT, które są określone podczas umieszczania komunikatu. W rezultacie kontekst tożsamości nie musi być powiązany z aplikacją, która pierwotnie umieściła komunikat, a kontekst źródłowy niekoniecznie odnosi się do aplikacji, która ostatnio umieściła komunikat; zależy to od projektu pakietu aplikacji.

Agent kanału komunikatów (MCA) nigdy nie zmienia kontekstu komunikatu. MCAs, które otrzymują komunikaty ze zdalnych menedżerów kolejek, używa opcji kontekstu MQPMO_SET_ALL_CONTEXT w wywołaniu MQPUT lub MQPUT1. Pozwala to odbierającemu agentowi MCA na zachowanie dokładnego kontekstu komunikatu, który podróżował z komunikatem od wysyłającego agenta MCA. Jednak wynikiem jest, że kontekst źródłowy nie jest powiązany z żadnym z MCAs, które wysłały i odbierają komunikat. Kontekst źródłowy odwołuje się do wcześniejszej aplikacji umieszczonej w komunikacie. Jeśli wszystkie aplikacje pośrednie przekazały kontekst komunikatu, kontekst źródłowy odwołuje się do samej aplikacji źródłowej.

W opisach pola kontekstu są opisane tak, jakby były używane w sposób opisany powyżej. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Pola dla deskryptora MQMD

Struktura MQMD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Tabela 54. Pola w strukturze MQMD		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Report</i>	Opcje dla komunikatów raportu	Raport
<i>MsgType</i>	Typ komunikatu	MsgType
<i>Expiry</i>	Czas życia komunikatu	MQMD-pole Expiry (MQMD)
<i>Feedback</i>	Informacja zwrotna lub kod przyczyny	MQMD-pole opinii (Feedback)
<i>Encoding</i>	Kodowanie numeryczne danych komunikatu	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków danych komunikatu	CodedCharSetId
<i>Format</i>	Nazwa formatu danych komunikatu	Formatowanie
<i>Priority</i>	Priorytet komunikatu	Priorytet
<i>Persistence</i>	Trwałość komunikatu	Trwałość

Tabela 54. Pola w strukturze MQMD (kontynuacja)

Pole	Opis	Temat
<i>MsgId</i>	Identyfikator komunikatu	MQMD-pole MsgId
<i>CorrelId</i>	Identyfikator korelacji	CorrelId
<i>BackoutCount</i>	Liczba wycofanych	BackoutCount
<i>ReplyToQ</i>	Nazwa kolejki odpowiedzi	Kolejka_zwrotna
<i>ReplyToQMGr</i>	Nazwa menedżera kolejek odpowiedzi	ReplyToQMGr
<i>UserIdentifier</i>	Identyfikator użytkownika	UserIdentifier
<i>AccountingToken</i>	Token rozliczania	AccountingToken
<i>ApplIdentityData</i>	Dane aplikacji odnoszące się do tożsamości	Dane_tożsamości_aplikacji
<i>PutApplType</i>	Typ aplikacji, która wstawiła komunikat	Typ_aplikacji_wstawiającej
<i>PutApplName</i>	Nazwa aplikacji umieszczonej w komunikacie.	Nazwa_aplikacji_wstawiającej
<i>PutDate</i>	Data umieszczenia komunikatu	PutDate
<i>PutTime</i>	Czas umieszczenia komunikatu	PutTime
<i>ApplOriginData</i>	Dane dotyczące wniosku dotyczące pochodzenia	Dane_pochodzenia_aplikacji
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQMD_VERSION_2.		
<i>GroupId</i>	Identyfikator grupy	GroupId
<i>MsgSeqNumber</i>	Numer kolejny komunikatu logicznego w grupie	Numer_kolejny_komunikatu
<i>Offset</i>	Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego	Depozycja
<i>MsgFlags</i>	Flagi komunikatu	MQMD-pole MsgFlags
<i>OriginalLength</i>	Długość oryginalnego komunikatu	OriginalLength

AccountingToken (MQBYTE32)

Jest to znacznik rozliczeniowy, część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatów zawiera sekcja [“Przegląd deskryptora MQMD”](#) na stronie 418 . Patrz także: [Kontekst komunikatu](#).

Produkt *AccountingToken* umożliwia odpowiednią aplikację do naliczania opłat za pracę wykonanego w wyniku komunikatu. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza jego zawartości.

Menedżer kolejek generuje następujące informacje w następujący sposób:

- Pierwszy bajt pola jest ustawiony na długość informacji rozliczeniowych znajdujących się w następujących bajtach. Długość ta mieści się w zakresie od zera do 30 i jest przechowywana w pierwszym bajcie jako binarna liczba całkowita.
- Drugi i kolejne bajty (określone w polu długości) są ustawiane na informacje rozliczeniowe odpowiednie dla środowiska.
 - W systemie z/OS informacje rozliczeniowe są ustawiane na:

- W przypadku zadania wsadowego z/OS -informacje rozliczeniowe z karty JES JOB lub z instrukcji JES ACCT na karcie EXEC (separatory przecinków są zmieniane na X'FF '). Informacje te są obcinane, jeśli to konieczne, do 31 bajtów.
- Dla TSO, numer konta użytkownika.
- W przypadku systemu CICSjednostka logiczna (LU 6.2) identyfikatora pracy (UEPUOWDS) (26 bajtów).
- W przypadku systemu IMS8-znakowa nazwa PSB konkatenowana z 16-znakowym znacznikiem odtwarzania IMS .
- W systemie IBM iinformacje rozliczeniowe są ustawiane na kod rozliczeniowy dla zadania.
- W systemie Systemy UNIXinformacje rozliczeniowe są ustawiane na liczbowy identyfikator użytkownika (w postaci znaków ASCII).
- W systemie Windowsinformacje rozliczeniowe są ustawiane na identyfikator zabezpieczeń systemu Windows (SID) w formacie skompresowanym. Identyfikator SID jednoznacznie identyfikuje identyfikator użytkownika zapisany w polu *UserIdentifier* . Jeśli identyfikator SID jest zapisany w polu *AccountingToken* , zostanie pominięty 6-bajtowy Urząd Identyfikatora (znajdujący się w trzecim i kolejnych bajtach identyfikatora SID). Na przykład, jeśli identyfikator SID Windows ma długość 28 bajtów, w polu *AccountingToken* zapisywane są 22 bajty informacji o identyfikatorze SID.
- Ostatni bajt (bajt 32) pola rozliczeniowego jest ustawiany na typ znacznika rozliczania (w tym przypadku MQACTT_NT_SECURITY_ID, x '0b'):

MQACTT_CICS_LUOW_ID

CICS Identyfikator LUOW.

MQACTT_NT_SECURITY_ID

Identyfikator zabezpieczeń produktu Windows .

MQACTT_OS400_ACCOUNT_TOKEN

IBM i token rozliczania.

MQACTT_UNIX_NUMERIC_ID

Identyfikator liczbowy Systemy UNIX .

MQACTT_USER,

Zdefiniowany przez użytkownika token rozliczania.

MQACTT_UNKNOWN

Nieznany typ znacznika rozliczania.

Typ znacznika rozliczania jest ustawiony na wartość jawną tylko w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami. W innych środowiskach typ znacznika rozliczania jest ustawiany na wartość MQACTT_UNKNOWN. W tych środowiskach należy użyć pola *PutApplType* , aby wywnioskować typ odebranego znacznika rozliczeniowego.

- Wszystkie pozostałe bajty są ustawione na zero binarne.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT został określony w parametrze *PutMsgOpts* . Jeśli nie zostanie podana opcja MQPMO_SET_IDENTITY_CONTEXT ani MQPMO_SET_ALL_CONTEXT, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *AccountingToken* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *AccountingToken* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis opcji MQPMO_RETAIN w produkcie “Opcje MQPMO (MQLONG)” na stronie 507 , aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest używana jako *AccountingToken* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *AccountingToken* we wszystkich publikacjach wysyłanych do tych subskrybentów. Jeśli komunikat nie ma kontekstu, to pole jest całkowicie binarne zero.

To jest pole wyjściowe dla wywołania MQGET.

To pole nie podlega żadnym tłumaczeniom opartym na zestawie znaków menedżera kolejek; pole jest traktowane jako łańcuch bitów, a nie jako łańcuch znaków.

Menedżer kolejek nie wykonuje żadnych informacji z informacjami w tym polu. Aplikacja musi interpretować informacje, jeśli chce korzystać z informacji do celów księgowych.

Dla pola *AccountingToken* można użyć następującej wartości specjalnej:

MQACT_NONE

Nie określono znacznika rozliczeniowego.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQACT_NONE_ARRAY; ma ona taką samą wartość jak MQACT_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową tego pola jest MQACT_NONE.

Dane ApplIdentity(MQCHAR32)

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i [Kontekst komunikatu](#).

ApplIdentityData to informacje, które są definiowane przez pakiet aplikacji i mogą być używane w celu udostępnienia dodatkowych informacji o komunikacie lub jego inicjatorze. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Gdy menedżer kolejek generuje te informacje, jest on całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT został określony w parametrze *PutMsgOpts*. Jeśli występuje znak o kodzie zero, wartość NULL i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli nie zostanie podana opcja MQPMO_SET_IDENTITY_CONTEXT ani MQPMO_SET_ALL_CONTEXT, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *ApplIdentityData*, który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *ApplIdentityData* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis opcji MQPMO_RETAIN, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest używana jako *ApplIdentityData*, gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *ApplIdentityData* we wszystkich publikacjach wysyłanych do tych subskrybentów. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez wartość MQ_APPL_IDENTITY_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 32 puste znaki w innych językach programowania.

ApplOrigin-dane (MQCHAR4)

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i [Kontekst komunikatu](#).

ApplOriginData to informacje zdefiniowane przez pakiet aplikacji, które mogą być używane do udostępniania dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiona przez aplikacje działające z odpowiednim uprawnieniem użytkownika w celu wskazania, czy dane tożsamości są zaufane.

Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Gdy menedżer kolejek generuje te informacje, jest on całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PutMsgOpts* podano wartość MQPMO_SET_ALL_CONTEXT. Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące znaki w puste znaki. Jeśli parametr MQPMO_SET_ALL_CONTEXT nie został określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez wartość MQ_APPL_ORIGIN_DATA_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C i 4 puste znaki w innych językach programowania.

Gdy komunikat jest publikowany, chociaż ustawiona jest wartość *ApplOriginDane*, oznacza to, że jest pusta w otrzymanej przez nią subskrypcji.

BackoutCount (MQLONG)

Jest to liczba określająca, ile razy komunikat został wcześniej zwrócony przez wywołanie MQGET jako część jednostki pracy, a następnie wycofał się z niego. Pomaga on w wykrywaniu błędów przetwarzania opartych na treści wiadomości. Liczba ta nie obejmuje wywołań MQGET, które określają dowolną z opcji MQGMO_BROWSE_*.

Dokładność tego licznika ma wpływ na atrybut kolejki *HardenGetBackout*; patrz [“Atrybuty dla kolejek” na stronie 845](#).

W systemie z/OS wartość 255 oznacza, że komunikat został wycofany z kopii zapasowej 255 lub więcej razy; zwrócona wartość nigdy nie jest większa niż 255.

To jest pole wyjściowe dla wywołania MQGET. Jest on ignorowany w przypadku wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest 0.

CodedCharSetId (MQLONG)

To pole określa identyfikator zestawu znaków dla danych znakowych w treści komunikatu.

Uwaga: Dane znakowe w strukturze MQMD i innych strukturach danych produktu MQ, które są parametrami wywołań, muszą znajdować się w zestawie znaków menedżera kolejek. Atrybut ten jest zdefiniowany przez atrybut *CodedCharSetId* menedżera kolejek. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla menedżera kolejek” na stronie 810](#).

Jeśli to pole jest ustawione na wartość MQCCSI_Q_MGR podczas wywoływania komendy MQGET z opcją MQGMO_CONVERT w opcjach, to zachowanie jest różne w przypadku aplikacji klienta i serwera. W przypadku aplikacji serwera strona kodowa używana do konwersji znaków to *CodedCharSetId* menedżera kolejek. W przypadku aplikacji klienckich strona kodowa używana do konwersji znaków jest bieżącą stroną kodową ustawień narodowych.

W przypadku aplikacji klienckich wartość MQCCSI_Q_MGR jest wypełniona w zależności od ustawień narodowych klienta, a nie od menedżera kolejek. Wyjątkiem od tej reguły jest umieszczanie komunikatu w kolejce mostu IMS; co jest zwracane w polu *CodedCharSetId* deskryptora MQMD, jest to identyfikator CCSID menedżera kolejek.

Nie można używać następującej wartości specjalnej:

MQCCSI_APPL

Powoduje to podanie niepoprawnej wartości w polu *CodedCharSetId* deskryptora MQMD i powoduje zwrócenie kodu powrotu MQRC_SOURCE_CCSD_ERROR (lub MQRC_FORMAT_ERROR w przypadku produktu z/OS). gdy komunikat jest odbierany za pomocą wywołania MQGET z opcją MQGMO_CONVERT.

Można użyć następujących wartości specjalnych:

MQCCSI_Q_MGR

Dane znakowe w komunikacie znajdują się w zestawie znaków menedżera kolejek.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia tę wartość w strukturze MQMD, która jest wysyłana z komunikatem do identyfikatora zestawu znaków true w menedżerze kolejek. W wyniku tego wartość MQCCSI_Q_MGR nigdy nie jest zwracana przez wywołanie MQGET.

MQCCSI_DEFAULT

CodedCharSetId danych w polu *String* jest definiowane przez pole *CodedCharSetId* w strukturze nagłówka poprzedzające strukturę MQCFH lub przez pole *CodedCharSetId* w strukturze MQMD, jeśli wartość MQCFH znajduje się na początku komunikatu.

MQCCSI_INHERIT

Dane znakowe w komunikacie znajdują się w tym samym zestawie znaków, co ta struktura. Jest to zestaw znaków menedżera kolejek. (Tylko dla MQMD: MQCCSI_INHERIT ma takie samo znaczenie jak MQCCSI_Q_MGR).

Menedżer kolejek zmienia tę wartość w strukturze MQMD, która jest wysyłana wraz z komunikatem do identyfikatora rzeczywistego zestawu znaków MQMD. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie należy używać wartości MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcie MQMD jest MQAT_BROKER.

MQCCSI_EMBEDDED

Dane znakowe w komunikacie znajdują się w zestawie znaków z identyfikatorem, który jest zawarty w samych danych komunikatu. Może istnieć dowolna liczba identyfikatorów zestawów znaków osadzonych w danych komunikatu, które dotyczą różnych części danych. Ta wartość musi być używana dla komunikatów PCF (w formacie MQFMT_ADMIN, MQFMT_EVENT lub MQFMT_PCF), które zawierają dane w mieszance zestawów znaków. Każda struktura MQCFST, MQCFSL i MQCFSF zawarta w komunikacie PCF musi mieć określony jawny identyfikator zestawu znaków, a nie MQCCSI_DEFAULT.

Jeśli komunikat o formacie MQFMT_EMBEDDED_PCF ma zawierać dane w mieszance zestawów znaków, nie należy używać komendy MQCCSI_EMBEDDED. Zamiast tego należy ustawić wartość MQEPH_CCSID_EMBEDDED w polu *Flags* w strukturze MQEPH. Jest to równoważne ustawieniu wartości MQCCSI_EMBEDDED w poprzedniej strukturze. Każda struktura MQCFST, MQCFSL i MQCFSF zawarta w komunikacie PCF musi mieć określony jawny identyfikator zestawu znaków, a nie MQCCSI_DEFAULT. Więcej informacji na temat struktury MQEPH zawiera sekcja [“MQEPH-osadzony nagłówek PCF” na stronie 362](#).

Tę wartość należy podać tylko w wywołaniach MQPUT i MQPUT1. Jeśli jest ona określona w wywołaniu MQGET, uniemożliwia ona konwersję komunikatu.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia wartości MQCCSI_Q_MGR i MQCCSI_INHERIT w strukturze MQMD, która jest wysyłana z komunikatem zgodnie z powyższym opisem, ale nie powoduje zmiany deskryptora MQMD określonego w wywołaniu MQPUT lub MQPUT1. Żadna inna kontrola nie jest przeprowadzana zgodnie z podaną wartością.

Aplikacje, które pobierają komunikaty, muszą porównać to pole z wartością oczekiwaną przez aplikację. Jeśli wartości różnią się, aplikacja może wymagać konwersji danych znakowych w komunikacie.

Jeśli w wywołaniu MQGET zostanie określona opcja MQGMO_CONVERT, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest identyfikatorem kodowanego zestawu znaków, do którego w razie potrzeby można przekształcić dane komunikatu. Jeśli konwersja jest pomyślna lub niepotrzebna, wartość ta pozostaje niezmienną (z tą różnicą, że wartość MQCCSI_Q_MGR lub MQCCSI_INHERIT jest przekształcana na wartość rzeczywistą). Jeśli konwersja nie powiedzie się, wartość podana po wywołaniu MQGET reprezentuje identyfikator kodowanego zestawu znaków dla nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W przeciwnym razie jest to pole wyjściowe wywołania MQGET, a także pole wejściowe dla wywołań MQPUT i MQPUT1. Początkowa wartość tego pola to MQCCSI_Q_MGR.

CorrelId (MQBYTE24)

Pole *CorrelId* jest właściwością w nagłówku komunikatu, która może być używana do identyfikowania konkretnego komunikatu lub grupy komunikatów.

Jest to łańcuch bajtowy, którego aplikacja może użyć do powiązania jednego komunikatu z innym, lub do powiązania komunikatu z innymi pracami wykonywanego przez aplikację. Identyfikator korelacji jest stałą właściwością komunikatu i utrzymuje się między restartami menedżera kolejek. Ponieważ

identyfikator korelacji jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator korelacji *nie* jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do drugiego.

W przypadku wywołań MQPUT i MQPUT1 aplikacja może określić dowolną wartość. Menedżer kolejek przesyła tę wartość wraz z komunikatem i dostarcza ją do aplikacji, która wysyła żądanie pobrania komunikatu.

Jeśli aplikacja określi wartość MQPMO_NEW_CORREL_ID, menedżer kolejek generuje unikalny identyfikator korelacji, który jest wysyłany z komunikatem, a także zwracany do aplikacji wysyłającej na wyjściu z wywołania MQPUT lub MQPUT1.

Identyfikator korelacji wygenerowany przez menedżer kolejek składa się z 3-bajтового identyfikatora produktu (AMQ lub CSQ w kodzie ASCII lub EBCDIC), po którym następuje jeden zarezerwowany bajt i implementacja specyficzna dla produktu w postaci unikalnego łańcucha. W produkcie IBM MQ ten łańcuch implementacji specyficznej dla produktu zawiera pierwsze 12 znaków nazwy menedżera kolejek oraz wartość uzyskana z zegara systemowego. Dlatego wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się od pierwszych 12 znaków, aby identyfikatory komunikatów były unikalne. Możliwość wygenerowania unikalnego łańcucha zależy również od tego, że zegar systemowy nie jest zmieniany wstecz. Aby wyeliminować możliwość utworzenia identyfikatora komunikatu wygenerowanego przez menedżer kolejek duplikujący wygenerowany przez aplikację, aplikacja musi unikać generowania identyfikatorów z początkowymi znakami z zakresu od A do I w kodzie ASCII lub EBCDIC (X'41 'do X'49' i X'C1'do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

Ten wygenerowany identyfikator korelacji jest przechowywany razem z komunikatem, jeśli jest on zachowywany, i jest używany jako identyfikator korelacji, gdy komunikat jest wysyłany jako publikacja do subskrybentów, którzy określają wartość MQCI_NONE w polu identyfikatora SubCorrelw zmaterializowanej tabeli MQSD przekazanej w wywołaniu MQSUB. Więcej informacji na temat zachowanych publikacji zawiera sekcja [Opcje MQPMO](#).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole *CorrelId* w sposób określony przez pole *Report* oryginalnego komunikatu, MQRO_COPY_MSG_ID_TO_CORREL_ID lub MQRO_PASS_CORREL_ID. Aplikacje, które generują komunikaty raportów, muszą również to zrobić.

W przypadku wywołania MQGET *CorrelId* jest jednym z pięciu pól, których można użyć do wybrania konkretnego komunikatu, który ma zostać pobrany z kolejki. Aby uzyskać szczegółowe informacje na temat określania wartości dla tego pola, należy zapoznać się z opisem pola *MsgId*.

Określenie parametru MQCI_NONE jako identyfikatora korelacji ma taki sam efekt, jak *nie* określenie wartości MQMO_MATCH_CORREL_ID, czyli *any* identyfikator korelacji zostanie dopasowany.

Jeśli opcja MQGMO_MSG_UNDER_CURSOR jest określona w parametrze *GetMsgOpts* w wywołaniu MQGET, to pole jest ignorowane.

W przypadku powrotu z wywołania MQGET pole *CorrelId* jest ustawione na identyfikator korelacji zwróconego komunikatu (jeśli istnieje).

Można użyć następujących wartości specjalnych:

MQCI_NONE

Nie określono identyfikatora korelacji.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest także stała MQCI_NONE_ARRAY; ta sama wartość ma taką samą wartość jak MQCI_NONE, ale jest tablicą znaków zamiast łańcucha.

MQCI_NOWA_SESJA

Komunikat jest początkiem nowej sesji.

Ta wartość jest rozpoznawana przez CICS bridge jako wskazującą początek nowej sesji, czyli początek nowej sekwencji komunikatów.

Dla języka programowania C jest również zdefiniowana stała zmienna MQCI_NEW_SESSION_ARRAY. Ma ona taką samą wartość jak MQCI_NEW_SESSION, ale jest tablicą znaków zamiast łańcucha.

W przypadku wywołania MQGET jest to pole wejściowe/wyjściowe. W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe, jeśli parametr MQPMO_NEW_CORREL_ID ma wartość *not* (nie określono) i pole wyjściowe, jeśli określono parametr MQPMO_NEW_CORREL_ID *is* . Długość tego pola jest podana przez wartość MQ_CORREL_ID_LENGTH. Wartością początkową tego pola jest MQCI_NONE.

Uwaga:

Nie można przekazać identyfikatora korelacji publikacji w hierarchii. Pole jest używane przez menedżer kolejek.

Kodowanie (MQLONG)

Określa kodowanie liczbowe danych liczbowych w komunikacie; nie ma zastosowania do danych liczbowych w samej strukturze MQMD. Kodowanie numeryczne definiuje reprezentację używaną dla binarnych liczb całkowitych, liczb całkowitych upakowanych liczb całkowitych i zmiennopozycyjnych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Zdefiniowane są następujące wartości specjalne:

MQENC_NATIVE

Kodowanie jest domyślne dla języka programowania i komputera, na którym działa aplikacja.

Uwaga: Wartość tej stałej zależy od języka programowania i środowiska. Z tego powodu aplikacje muszą być kompilowane za pomocą plików nagłówkowych, makro, COPY lub INCLUDE odpowiednich dla środowiska, w którym aplikacja będzie uruchamiana.

Aplikacje, które umieszczają komunikaty zwykle określają parametr MQENC_NATIVE. Aplikacje, które pobierają komunikaty, muszą porównać to pole z wartością MQENC_NATIVE; jeśli wartości różnią się, aplikacja może wymagać konwersji danych liczbowych w komunikacie. Użyj opcji MQGMO_CONVERT, aby zażądać, aby menedżer kolejek przekształcić komunikat w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat konstruowania pola *Encoding* zawiera sekcja [“Kodowanie komputera” na stronie 912](#) .

Jeśli w wywołaniu MQGET zostanie określona opcja MQGMO_CONVERT, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest kodowaniem, do którego w razie potrzeby można przekształcić dane komunikatu. Jeśli konwersja jest pomyślna lub niepotrzebna, wartość ta pozostaje niezmienną. Jeśli konwersja nie powiedzie się, wartość podana po wywołaniu MQGET reprezentuje kodowanie nieprzekształconego komunikatu, które jest zwracane do aplikacji.

W innych przypadkach jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQENC_NATIVE.

Utrata ważności (MQLONG)

Jest to okres wyrażony w dziesiątych częściach sekundy, ustawiany przez aplikację, która umieszcza komunikat. Komunikat zostaje zakwalifikowany do usunięcia, jeśli nie został usunięty z kolejki docelowej przed upływem tego okresu.

Na przykład, aby ustawić jedną minutę na czas utraty ważności, należy ustawić wartość **MQMD.Expiry** do 600.

Wartość ta jest zmniejszana, tak aby odzwierciedlała czas, przez jaki komunikat jest wyświetlany w kolejce docelowej, a także w pośrednich kolejkach transmisji, jeśli jest on do kolejki zdalnej. Można je również zmniejszać za pomocą agentów kanałów komunikatów, aby odzwierciedlały czasy transmisji, jeśli są one istotne. Podobnie, aplikacja przekazując ten komunikat do innej kolejki może zmniejszyć wartość, jeśli jest to konieczne, o ile zachowała ona komunikat przez istotny czas. Czas utraty ważności jest jednak traktowany jako przybliżony, a wartość nie musi być zmniejszana, aby odzwierciedlać małe przedziały czasu.

Gdy komunikat jest pobierany przez aplikację przy użyciu wywołania MQGET, pole *Expiry* reprezentuje czas utraty ważności, który nadal pozostaje.

Po upływie czasu utraty ważności komunikatu, staje się on zakwalifikowany do odrzucenia przez menedżer kolejek. Komunikat jest odrzucany, gdy wystąpi wywołanie funkcji MQGET z przeglądaniem lub nieprzeglądaniem, które zwróciłoby komunikat, gdyby nie utracił on już ważności. Na przykład nieprzeglądanie wywołania MQGET z polem *MatchOptions* w zestawie MQGMO ustawionym na MQMO_NONE z kolejki uporządkowanej FIFO odrzuci wszystkie przedawnione komunikaty aż do pierwszego nieprzedawnionego komunikatu. W przypadku kolejki uporządkowanej według priorytetu to samo wywołanie odrzuci przedawnione komunikaty o wyższym priorytecie i komunikaty o równym priorytecie, które dotarły do kolejki przed pierwszym nieprzeterminowanym komunikatem.

Komunikat, który utracił ważność, nigdy nie jest zwracany do aplikacji (przy użyciu przeglądania lub wywołania MQGET bez przeglądania), więc wartość w polu *Expiry* deskryptora komunikatu po pomyślnym wywołaniu MQGET jest większa od zera lub wartość specjalna MQEI_UNLIMITED.

Jeśli komunikat jest umieszczany w kolejce zdalnej, komunikat może tracić ważność (i być odrzucany), gdy znajduje się on w pośredniej kolejce transmisji, zanim komunikat osiągnie kolejkę docelową.

Raport jest generowany, gdy przedawniony komunikat jest odrzucany, jeśli w komunikacie określono jedną z opcji raportu MQRO_EXPIRATION_*. Jeśli żadna z tych opcji nie zostanie określona, taki raport nie zostanie wygenerowany; zakłada się, że komunikat nie będzie już istotny po tym okresie (być może dlatego, że później został zastąpiony przez komunikat).

W przypadku komunikatu umieszczonego w punkcie synchronizacji przedział czasu utraty ważności rozpoczyna się od momentu umieszczenia komunikatu, a nie czasu, w którym następuje zatwierdzenie punktu synchronizacji. Możliwe jest, że przedział czasu utraty ważności może zostać przekazany przed zatwierdzonym punktem synchronizacji. W tym przypadku komunikat zostanie usunięty po pewnym czasie po operacji zatwierdzania, a komunikat nie zostanie zwrócony do aplikacji w odpowiedzi na operację MQGET.

Każdy inny program, który usuwa komunikaty na podstawie czasu utraty ważności, musi również wysłać odpowiedni komunikat raportu, jeśli zażądano jednego z nich.

Uwagi:

1. Jeśli komunikat jest umieszczany z czasem *Expiry* o wartości zero lub o numerze większym niż 999 999 999, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem i kodem przyczyny MQRC_EXPIRY_ERROR; w tym przypadku nie jest generowany żaden komunikat raportu.

Aby włączyć kod przyczyny 2013, MQRC_EXPIRY_ERROR, należy włączyć zmienną środowiskową AMQ_ENFORCE_MAX_EXPIRY_ERROR.

Poniżej przedstawiono przykład dla produktu Linux:

```
$ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
```

Należy pamiętać, że:

- Ważne jest, aby wyeksportować zmienną
 - Wartość rzeczywista jest ignorowana, jednak podczas przeglądania konfiguracji przy użyciu składnika True może być przydatne.
2. Ponieważ komunikat z czasem utraty ważności, który upłynął, może nie zostać usunięty do czasu późniejszego, mogą istnieć komunikaty w kolejce, które przeszły upływ czasu utraty ważności, a więc nie kwalifikują się do pobrania. Komunikaty te liczą się jednak w kierunku liczby komunikatów w kolejce dla wszystkich celów, w tym dla wyzwalania głębokości.
 3. Raport o utracie ważności jest generowany, jeśli zażądano, gdy komunikat jest odrzucany, a nie w momencie, gdy staje się on uprawniony do usunięcia.
 4. Odrzucanie przedawnionego komunikatu i generowanie raportu o utracie ważności, jeśli zażądano, nigdy nie są częścią jednostki pracy aplikacji, nawet jeśli komunikat został zaplanowany do usunięcia w wyniku wywołania MQGET działającego w ramach jednostki pracy.
 5. Jeśli komunikat o prawie ważności jest pobierany za pomocą wywołania MQGET w ramach jednostki pracy, a następnie wycofana jest jednostka pracy, może on zostać zakwalifikowany do usunięcia, zanim będzie można go ponownie pobrać.

6. Jeśli komunikat o prawie ważności jest zablokowany przez wywołanie MQGET z parametrem MQGMO_LOCK, może on zostać zakwalifikowany do usunięcia, zanim będzie mógł zostać pobrany przez wywołanie MQGET z kodem przyczyny MQGMO_MSG_UNDER_CURSOR;. W tym przypadku zwracany jest kod przyczyny MQRC_NO_MSG_UNDER_CURSOR. Jeśli to nastąpi, zostanie zwrócony kod przyczyny.

7. Po pobraniu komunikatu z żądaniem o czasie utraty ważności większym niż zero aplikacja może wykonać jedno z następujących działań, gdy wyśle komunikat odpowiedzi:

- Skopiuj pozostały czas utraty ważności z komunikatu żądania do komunikatu odpowiedzi.
- Ustaw czas utraty ważności w komunikacie odpowiedzi na wartość jawną większą niż zero.
- Ustaw czas utraty ważności w komunikacie odpowiedzi na MQEI_UNLIMITED.

Działanie, które ma być podjęte, zależy od projektu aplikacji. Jednak domyślnym działaniem umieszczania komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) musi być zachowanie pozostałego czasu utraty ważności komunikatu, a także kontynuowanie jego zmniejszania.

8. Komunikaty wyzwacza są zawsze generowane z MQEI_UNLIMITED.

9. Komunikat (zwykle w kolejce transmisji), który ma nazwę *Format* MQFMT_XMIT_Q_HEADER, ma drugi deskryptor komunikatu w tabeli MQXQH. Z tego powodu są powiązane z nim dwa pola *Expiry*. W tym przypadku należy odnotować następujące dodatkowe punkty:

- Gdy aplikacja umieszcza komunikat w kolejce zdalnej, menedżer kolejek umieszcza komunikat początkowo w lokalnej kolejce transmisji, a następnie prefikuje dane komunikatu aplikacji ze strukturą MQXQH. Menedżer kolejek ustawia wartości dwóch pól programu *Expiry* tak, aby były takie same jak wartości określone przez aplikację.

Jeśli aplikacja umieszcza komunikat bezpośrednio w lokalnej kolejce transmisji, dane komunikatu muszą już zaczynać się od struktury MQXQH, a nazwa formatu musi być nazwą MQFMT_XMIT_Q_HEADER. W takim przypadku aplikacja nie musi ustawiać wartości tych dwóch pól *Expiry*, aby były takie same. (Menedżer kolejek sprawdza, czy pole *Expiry* w tabeli MQXQH zawiera poprawną wartość, a także czy dane komunikatu są wystarczająco długie, aby można je było dołączyć). W przypadku aplikacji, która może zapisywać bezpośrednio do kolejki transmisji, aplikacja musi utworzyć nagłówek kolejki transmisji z osadzonym deskryptorem komunikatu. Jeśli jednak wartość utraty ważności w deskrytorze komunikatu zapisanym do kolejki transmisji jest niespójna z wartością w osadzonym deskrytorze komunikatu, następuje odrzucenie błędu utraty ważności.

- Gdy komunikat o nazwie *Format* MQFMT_XMIT_Q_HEADER jest pobierany z kolejki (niezależnie od tego, czy jest to kolejka normalna, czy kolejka transmisji), menedżer kolejek zmniejsza *oba* te pola *Expiry* z czasem spędzonym na oczekiwaniu w kolejce. Jeśli dane komunikatu nie są wystarczająco długie, aby dołączyć pole *Expiry* w tabeli MQXQH, nie jest zgłaszany żaden błąd.
- Menedżer kolejek używa pola *Expiry* w oddzielnym deskrytorze komunikatu (to znaczy nie jest to element w deskrytorze komunikatu osadzonym w strukturze MQXQH) w celu sprawdzenia, czy komunikat jest zakwalifikowany do usunięcia.
- Jeśli początkowe wartości dwóch pól *Expiry* są różne, czas *Expiry* w oddzielnym deskrytorze komunikatu, gdy komunikat jest pobierany, może być większy od zera (tak więc komunikat nie kwalifikuje się do usunięcia), podczas gdy czas zgodny z polem *Expiry* w tabeli MQXQH upłynie. W tym przypadku pole *Expiry* w tabeli MQXQH jest ustawione na zero.

10. Czas utraty ważności dla komunikatu odpowiedzi zwróconego z mostu IMS jest nieograniczony, chyba że w polu Flags w tabeli MQIIH ustawiono wartość MQIIH_PASS_EXPIRATION. Więcej informacji na ten temat zawiera sekcja [Flagi](#).

Rozpoznawana jest następująca wartość specjalna:

MQEI_UNLIMITED

Czas utraty ważności komunikatu jest nieograniczony.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Początkowa wartość tego pola to MQEI_UNLIMITED.

Wygaste komunikaty w systemie z/OS

W systemie IBM MQ for z/OS komunikaty, które utraciły ważność, są odrzucane przez następne odpowiednie wywołanie MQGET.

Jeśli jednak takie wywołanie nie zostanie wykonane, komunikat, który utracił ważność, nie jest usuwany, a w przypadku niektórych kolejek może być kumulowana duża liczba przedawnionych komunikatów. Aby temu zaradzić, należy ustawić menedżera kolejek w celu okresowego skanowania kolejek i odrzucania komunikatów, które utraciły ważność, w jednej lub kilku kolejkach, w jeden z następujących sposobów:

Skanowanie okresowe

Istnieje możliwość określenia okresu przy użyciu atrybutu menedżera kolejek EXPRYINT (okres ważności). Za każdym razem, gdy upłynie odstęp czasu ważności, menedżer kolejek wyszukuje kolejki kandydujące, które są warte skanowania, aby usunąć wygaste komunikaty.

Menedżer kolejek przechowuje informacje na temat komunikatów, które utraciły ważność w każdej kolejce, i wie, czy skanowanie przedawnionych komunikatów jest warte zachodu. Tak więc, tylko wybór kolejek jest skanowany w dowolnym momencie.

Kolejki współużytkowane są skanowane przez tylko jednego menedżera kolejek w grupie współużytkowania kolejki. Zwykle jest to pierwszy menedżer kolejek do zrestartowania, lub pierwszy, który ma ustawioną wartość EXPRYINT. Jeśli ten menedżer kolejek zakończy działanie, inny menedżer kolejek w grupie współużytkującej kolejkę przejmuje skanowanie kolejki. Ustaw tę samą wartość dla wszystkich menedżerów kolejek w grupie współużytkującej kolejkę i ustaw ją na tę samą wartość.

Należy pamiętać, że przetwarzanie utraty ważności jest wymagane dla każdej kolejki po restarcie menedżera kolejek, niezależnie od ustawienia EXPRYINT.

Żądanie jawne

Wydadaj komendę REFRESH QMGR TYPE (WAŻNOŚCI), określając kolejkę lub kolejki, które mają być skanowane.

Wymuszanie stosowania niższych czasów utraty ważności

Administratorzy mogą ograniczyć czas utraty ważności dowolnego komunikatu umieszczonego w kolejce lub tematu za pomocą atrybutu CAPEXPY określonego w atrybucie **CUSTOM** w kolejce lub w temacie.

 Aby włączyć tę funkcję w systemie z/OS, należy zastosować poprawkę APAR PI50761 i włączyć nową funkcję za pomocą komendy MQSC **RECOVER QMGR (TUNE CAPEXPY ON)**.

Ważne: Komenda **RECOVER QMGR** jest zarezerwowana do użycia przez produkt IBM i powinna być normalnie używana wyłącznie pod kierunkiem obsługi produktu IBM.

 Komendę można wprowadzić w konsoli lub w konfiguracji menedżera kolejek, dodając do pliku przetwarzanego w konkatencji CSQINP2 podczas uruchamiania menedżera kolejek. Należy zauważyć, że efekt działania komendy nie jest utrwalany w czasie restartu menedżera kolejek.

Czas utraty ważności określony w polu **Expiry** deskryptora MQMD przez aplikację, która jest większa niż wartość CAPEXPY określona w atrybucie **CUSTOM** w kolejce lub temacie, zostanie zastąpiona przez wartość CAPEXPY. Użyty zostanie czas utraty ważności określony przez aplikację, która jest niższa niż wartość CAPEXPY.

Należy zauważyć, że wartość **CAPEXPY** wyrażona jest w dziesiątych częściach sekundy, więc jedna minuta ma wartość 600.

Jeśli w ścieżce rozstrzygania jest używany więcej niż jeden obiekt, na przykład gdy komunikat jest umieszczany w aliasie lub w kolejce zdalnej, to najniższy ze wszystkich wartości CAPEXPY jest używany jako górny limit dla utraty ważności komunikatu.

Zmiany wartości CAPEXPY są wprowadzane natychmiast. Wartość utraty ważności jest wartościowana dla każdego umieszczonego w kolejce lub tematu i dlatego jest ona wrażliwa na rozdzielczość obiektu, która może być różna dla każdej operacji put.

Należy jednak pamiętać, że istniejące komunikaty w kolejce przed zmianą w **CAPEXP**RY nie mają wpływu na zmianę (oznacza to, że ich czas utraty ważności pozostaje nienaruszony). Tylko nowe komunikaty umieszczane w kolejce po zmianie w programie **CAPEXP**RY mają nowy czas utraty ważności.

Na przykład w klastrze, w którym operacja put jest wykonywana w kolejce otwartej za pomocą komendy *MQOO_BIND_NOT_FIXED*, komunikaty mogą mieć przypisane różne wartości utraty ważności dla każdego z nich, w zależności od wartości *CAPEXP*RY ustawionej dla kolejki transmisji, używanej przez kanał, która wysyła komunikat do wybranego docelowego menedżera kolejek.

Należy pamiętać, że wprowadzenie do kolejki lub tematu przez aplikację JMS, w której określono opóźnienie dostarczenia, kończy się niepowodzeniem z błędem *MQRC_EXPIRY_ERROR*, jeśli opóźnienie dostarczenia jest poza rozstrzygniętym czasem utraty ważności dla kolejki docelowej lub tematu. A *CAPEXP*RY attribute set on a queue resolved for a JMS Destination can cause this error.

Uwaga: Produkt **CAPEXP**RY nie może być używany w żadnych kolejkach, w których będą przechowywane komunikaty generowane przez produkt IBM MQ, takie jak dowolny system *SYSTEM.CLUSTER.** w kolejce i w systemie *SYSTEM.PROTECTION.POLICY.QUEUE*.

Opinia (MQLONG)

Pole Feedback jest używane z komunikatem typu *MQMT_REPORT* w celu wskazania rodzaju raportu i ma znaczenie tylko w przypadku tego typu komunikatu.

Pole może zawierać jedną z wartości *MQFB_** lub jedną z wartości *MQRC_**. Kody opinii są pogrupowane w następujący sposób:

MQFB_NONE

Nie podano informacji zwrotnych.

MQFB_SYSTEM_FIRST

Najniższa wartość dla informacji zwrotnych generowanych przez system.

MQFB_SYSTEM_LAST

Najwyższa wartość dla informacji zwrotnych generowanych przez system.

Zakres kodów sprzężenia zwrotnego generowanych przez system *MQFB_SYSTEM_FIRST* za pomocą komendy *MQFB_SYSTEM_LAST* zawiera ogólne kody opinii wymienione w tym temacie (*MQFB_**), a także kody przyczyny (*MQRC_**), które mogą wystąpić, gdy komunikat nie może zostać umieszczony w kolejce docelowej.

MQFB_APPL_FIRST

Najniższa wartość dla informacji zwrotnych generowanych przez aplikację.

MQFB_APPL_LAST

Najwyższa wartość dla informacji zwrotnych generowanych przez aplikację.

Aplikacje, które generują komunikaty raportów, nie mogą używać kodów opinii w zakresie systemowym (innym niż *MQFB_QUIT*), chyba że chcą symulować komunikaty raportów wygenerowane przez menedżer kolejek lub agenta kanału komunikatów.

W wywołaniach *MQPUT* lub *MQPUT1* podana wartość musi mieć wartość *MQFB_NONE* lub musi być w zakresie systemowym lub w zakresie aplikacji. Ta opcja jest sprawdzana niezależnie od wartości parametru *MsgType*.

Ogólne kody opinii:

MQFB_COA

Potwierdzenie przybycia do kolejki docelowej (patrz *MQRO_COA*).

MQFB_COD

Potwierdzenie dostarczenia do aplikacji odbierającej (patrz *MQRO_COD*).

MQFB_EXPIRATION

Komunikat został odrzucony, ponieważ nie został usunięty z kolejki docelowej przed upływem czasu jego utraty ważności.

MQFB_PAN

Powiadomienie o działaniu pozytywnym (patrz *MQRO_PAN*).

MQFB_NAN

Powiadomienie o działaniu negatywnym (patrz MQRO_NAN).

MQFB_QUIT

Zakończenie aplikacji.

Może to być używane przez program do planowania obciążenia w celu kontrolowania liczby działających instancji programu użytkowego. Wysłanie komunikatu MQMT_REPORT z tym kodem sprzężenia zwrotnego do instancji programu użytkowego wskazuje na tę instancję, że powinna ona zatrzymać przetwarzanie. Jednak stosowanie tej konwencji jest kwestią dla aplikacji. Nie jest ona wymuszana przez menedżer kolejek.

Kody sprzężenia zwrotnego kanału:

MQFB_CHANNEL_COMPLETED

Kanał został zakończony normalnie.

MQFB_CHANNEL_FAIL

Kanał został zakończony nieprawidłowo i przechodzi do stanu STOPPED.

MQFB_CHANNEL_FAIL_RETRY

Kanał został nieprawidłowo zakończony i przechodzi w stan RETRY.

IMS-kody sprzężenia zwrotnego mostu

Te kody są używane, gdy odebrano nieoczekiwany kod rozpoznania IMS-OTMA. Kod rozpoznania lub kod przyczyny 0x1A, kod przyczyny powiązany z tym kodem rozpoznania, jest wskazany w *Opisie zwrotnej*.

1. W przypadku kodów *Feedback* z zakresu MQFB_IMS_FIRST (300) za pomocą MQFB_IMS_LAST (399), odebrano kod rozpoznania inny niż 0x1A. *Kod rozpoznania* jest nadawany przez wyrażenie (*Opinia* - MQFB_IMS_FIRST+1)
2. W przypadku kodów *Feedback* z zakresu MQFB_IMS_NACK_1A_REASON_FIRST (600) za pomocą komendy MQFB_IMS_NACK_1A_REASON_LAST (855) otrzymano kod rozpoznania 0x1A. *Kod przyczyny* powiązany z kodem rozpoznania jest nadawany przez wyrażenie (*Opinia* - MQFB_IMS_NACK_1A_REASON_FIRST)

Znaczenie kodów rozpoznania IMS-OTMA i odpowiednich kodów przyczyny jest opisane w publikacji *Open Transaction Manager Access Guide and Reference*.

Most IMS może generować następujące kody sprzężenia zwrotnego:

MQFB_DATA_LENGTH_ZERO

Długość segmentu była równa zero w danych aplikacji komunikatu.

MQFB_DŁUGOŚĆ_DŁUGOŚĆ_UJEMNEGO

Długość segmentu była ujemna w danych aplikacji komunikatu.

MQFB_DATA_LENGTH_TOO_BIG

Długość segmentu była zbyt duża w danych aplikacji komunikatu.

MQFB_BUFFER_OVERFLOW

Wartość jednego z pól o długości spowodowałaaby przepełnienie buforu komunikatów.

MQFB_LENGTH_OFF_BY_ONE

Wartość jednego z pól długości była za krótka 1 bajt.

BŁĄD MQFB_IIH_ERROR

Pole *Format* w strukturze MQMD określa wartość MQFMT_IMS, ale komunikat nie rozpoczyna się od poprawnej struktury MQIIH.

MQFB_NOT_AUTHORIZED_FOR_IMS

Identyfikator użytkownika zawarty w deskrytorze komunikatu MQMD lub hasło zawarte w polu *Authenticator* w strukturze MQIIH nie powiodło się podczas sprawdzania poprawności, które zostało wykonane przez most IMS. W wyniku tego komunikat nie został przekazany do produktu IMS.

BŁĄD MQFB_IMS_ERROR

Program IMSzwrócił nieoczekiwany błąd. Zapoznaj się z dziennikiem błędów systemu IBM MQ w systemie, w którym znajduje się most IMS, aby uzyskać więcej informacji na temat tego błędu.

MQFB_IMS_FIRST

Jeśli kod rozpoznania IMS-OTMA nie jest kodem 0x1A, kody zwrotne wygenerowane przez produkt IMSSą w zakresie MQFB_IMS_FIRST (300) za pomocą MQFB_IMS_LAST (399). Kod rozpoznania IMS-OTMA sam w sobie to *Feedback* minus MQFB_IMS_ERROR.

MQFB_IMS_LAST

Najwyższa wartość dla informacji zwrotnych generowanych przez produkt IMS, gdy kod rozpoznania nie jest 0x1A.

MQFB_IMS_NACK_1A_REASON_FIRST

Gdy kod rozpoznania ma wartość 0x1A, kody zwrotne wygenerowane przez produkt IMSSą w zakresie MQFB_IMS_NACK_1A_REASON_FIRST (600) za pomocą komendy MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST

Najwyższa wartość dla informacji zwrotnych generowanych przez produkt IMS, gdy kod rozpoznania ma wartość 0x1A

Kody sprzężenia zwrotnego CICS-most: Następujące kody zwrotne mogą być generowane przez CICS bridge:

MQFB_CICS_APPL_ABENDED

Program użytkowy podany w komunikacie został nieprawidłowo zakończony. Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_APPL_NOT_STARTED

Działanie EXEC CICS LINK dla programu użytkowego określonego w komunikacie nie powiodło się. Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_NIEPOWODZENIE bridge_failure

Działanie CICS bridge zostało zakończone nieprawidłowo, bez zakończenia normalnego przetwarzania błędów.

MQFB_CICS-BŁĄD ID_CCSID

Niepoprawny identyfikator zestawu znaków.

MQFB_CICS-BŁĄD CIH_ERROR

Brak struktury nagłówka informacji CICS lub jest ona niepoprawna.

MQFB_CICS_COMMAREA_ERROR

Długość komendy CICS COMMAREA nie jest poprawna.

MQFB_CICS_CORREL_ID_ERROR

Niepoprawny identyfikator korelacji.

MQFB_CICS_DLQ_ERROR

Zadanie CICS bridge nie było w stanie skopiować odpowiedzi na to żądanie do kolejki niedostarczonych komunikatów. Żądanie zostało wycofane.

MQFB_CICS-BŁĄD __ENCODING_ERROR

Kodowanie jest niepoprawne.

MQFB_CICS_INTERNAL_ERROR

Program CICS bridge napotkał nieoczekiwany błąd.

Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_NOT_AUTHORIZED

Nieautoryzowany identyfikator użytkownika lub hasło nie jest poprawne.

Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_UOW_BACKED_OUT

Kopia zapasowa jednostki pracy została wykonana z jednego z następujących powodów:

- Wykryto awarię podczas przetwarzania innego żądania w ramach tej samej jednostki pracy.
- Abend CICS wystąpił, gdy jednostka pracy była w toku.

MQFB_CICS_UOW_ERROR

Pole elementu sterującego jednostki pracy *UOWControl* jest niepoprawne.

Kody informacji zwrotnych komunikatów trasy śledzenia:

MQFB_ACTIVITY,

Używany z formatem MQFMT_EMBEDDED_PCF, aby umożliwić użycie opcji danych użytkownika po raportach aktywności.

DZIAŁANIA MQFB_MAX_ACTIVITIES

Zwracane, gdy komunikat trasy śledzenia jest odrzucany, ponieważ liczba działań, w których uczestniczyli komunikat, przekracza limit maksymalnej aktywności.

MQFB_NOT_FORWARDED

Zwracane, gdy komunikat trasy śledzenia jest odrzucany, ponieważ ma zostać wysłany do zdalnego menedżera kolejek, który nie obsługuje komunikatów trasy śledzenia.

MQFB_NOT_DELIVERED

Zwracane, gdy komunikat trasy śledzenia jest odrzucany, ponieważ ma zostać umieszczony w kolejce lokalnej.

MQFB_UNSUPPORTED_FORWARDING

Zwracana, gdy komunikat trasy śledzenia jest odrzucany, ponieważ wartość w parametrze przekazywania jest nierozpoznana i znajduje się w odrzuconej masce bitowej.

MQFB_UNSUPPORTED_DELIVERY

Zwracany wówczas, gdy komunikat trasy śledzenia jest odrzucany, ponieważ wartość w parametrze dostarczania jest nierozpoznana i znajduje się w odrzuconej masce bitowej.

Kody przyczyny produktu IBM MQ: w przypadku komunikatów o wyjątkach produkt *Feedback* zawiera kod przyczyny produktu IBM MQ. Możliwe są następujące kody przyczyny:

MQRC_PUT_INHIBITED

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki.

MQRC_Q_FULL

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

MQRC_NOT_AUTHORIZED

(2035, X'7F3 ') Brak uprawnień do dostępu.

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF ') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE ') Długość komunikatu jest większa niż maksymalna dla kolejki.

Pełną listę kodów przyczyny można znaleźć w:

- Informacje na temat produktu IBM MQ for z/OS zawiera sekcja [Kody zakończenia i kody przyczyny funkcji API](#).
- Informacje na temat wszystkich innych platform zawiera sekcja [Kody zakończenia i przyczyny interfejsu API](#).

.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest MQFB_NONE.

Format (MQCHAR8)

Jest to nazwa, która jest używana przez nadawcę wiadomości do wskazania odbiorcy charakteru danych w komunikacie. Dla nazwy można określić dowolne znaki znajdujące się w zestawie znaków menedżera kolejek, ale należy ograniczyć nazwę do następujących znaków:

- Wielkie litery od A do Z
- Cyfry od 0 do 9

Jeśli używane są inne znaki, tłumaczenie nazwy między zestawami znaków wysyłających i odbierających menedżerów kolejek może nie być możliwe.

Należy dopełniać nazwę spacjami na długość pola lub użyć znaku o kodzie zero, aby zakończyć nazwę przed końcem pola. NULL i kolejne znaki są traktowane jako znaki puste. Nie należy określać nazwy z odstępami wiodącymi ani odstępami osadzonymi. W przypadku wywołania MQGET menedżer kolejek zwraca nazwę dopełnioną spacjami do długości pola.

Menedżer kolejek nie sprawdza, czy nazwa jest zgodna z zaleceniami opisanymi powyżej.

Nazwy rozpoczynające się od MQ w wielkich, dolnych i mieszanych przypadkach mają znaczenie, które są definiowane przez menedżer kolejek; nie należy używać nazw rozpoczynających się od tych liter dla własnych formatów. Wbudowane formaty menedżera kolejek to:

MQFMT_NONE

Rodzaj danych nie jest zdefiniowany: dane nie mogą być przekształcane, gdy komunikat jest pobierany z kolejki za pomocą opcji MQGMO_CONVERT.

Jeśli w wywołaniu MQGET zostanie określona wartość MQGMO_CONVERT, a zestaw znaków lub kodowanie danych w komunikacie różni się od wartości określonej w parametrze *MsgDesc*, to komunikat zostanie zwrócony z następującymi kodami zakończenia i przyczyny (przy założeniu, że nie wystąpiły inne błędy):

- Kod zakończenia MQCC_WARNING i kod przyczyny MQRC_FORMAT_ERROR, jeśli na początku komunikatu znajduje się dane MQFMT_NONE.
- Kod zakończenia MQCC_OK i kod przyczyny MQRC_NONE, jeśli na końcu komunikatu znajduje się dane MQFMT_NONE (to znaczy poprzedzone co najmniej jedną strukturą nagłówka MQ). Struktury nagłówka MQ są przekształcane w żądany zestaw znaków i kodowanie w tym przypadku.

W przypadku języka programowania C zdefiniowana jest również stała MQFMT_NONE_ARRAY; ta wartość ma taką samą wartość jak MQFMT_NONE, ale jest tablicą znaków zamiast łańcucha.

ADMINISTRATOR MQFMT_ADMIN

Komunikat jest komunikatem żądania lub odpowiedzi serwera komend w formacie programu programowalnego (PCF). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT. Więcej informacji na temat używania programowalnych komunikatów formatu komend zawiera sekcja [Korzystanie z formatów komend programowalnych](#).

W przypadku języka programowania C zdefiniowana jest również stała MQFMT_ADMIN_ARRAY. Ma ona taką samą wartość jak MQFMT_ADMIN, ale jest tablicą znaków zamiast łańcucha.

MQFMT_CICS

Dane komunikatu rozpoczynają się od nagłówka informacyjnego produktu CICS MQCIH, po którym następuje dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *Format* w strukturze MQCIH.

W systemie z/OS należy określić opcję MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić komunikaty o formacie MQFMT_CICS.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_CICS_ARRAY; ta wartość ma taką samą wartość jak MQFMT_CICS, ale jest tablicą znaków zamiast łańcucha.

MQFMT_COMMAND_1

Komunikat to komunikat odpowiedzi serwera komend MQSC, zawierający liczbę obiektów, kod zakończenia i kod przyczyny. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_COMMAND_1_ARRAY . Ma ona taką samą wartość jak MQFMT_COMMAND_1, ale jest tablicą znaków zamiast łańcucha.

MQFMT_COMMAND_2

Komunikat jest komunikatem odpowiedzi serwera komend MQSC, zawierającym informacje na temat żądanych obiektów. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_COMMAND_2_ARRAY . Ma ona taką samą wartość co MQFMT_COMMAND_2, ale jest tablicą znaków zamiast łańcucha.

MQFMT_DEAD_LETTER_HEADER

Dane komunikatu rozpoczynają się od nagłówka niedostarczonych komunikatów MQDLH. Dane z oryginalnego komunikatu są natychmiast następujące po strukturze MQDLH. Nazwa formatu oryginalnych danych komunikatu jest podawana przez pole *Format* w strukturze MQDLH; szczegółowe informacje na temat tej struktury można znaleźć w sekcji [“MQDLH-nagłówki Dead-letter”](#) na stronie 349 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Raporty COA i COD nie są generowane dla komunikatów, które mają *Format* o wartości MQFMT_DEAD_LETTER_HEADER.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_DEAD_LETTER_HEADER_ARRAY; ta wartość ma taką samą wartość jak MQFMT_DEAD_LETTER_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_DIST_HEADER

Dane komunikatu rozpoczynają się od nagłówka listy dystrybucyjnej MQDH; obejmuje to tablice rekordów MQOR i MQPMR. Po nagłówku listy dystrybucyjnej mogą następować dodatkowe dane. Format dodatkowych danych (jeśli istnieje) jest podany w polu *Format* w strukturze MQDH. Szczegółowe informacje na temat tej struktury zawiera sekcja [“MQDH-nagłówki dystrybucji”](#) na stronie 343 . Komunikaty o formacie MQFMT_DIST_HEADER mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Ten format jest obsługiwany w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_DIST_HEADER_ARRAY; ma ona taką samą wartość co MQFMT_DIST_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_EMBEDDED_PCF

Format komunikatu trasy śledzenia, pod warunkiem, że wartość komendy PCF jest ustawiona na MQCMD_TRACE_ROUTE. Użycie tego formatu pozwala na wysyłanie danych użytkownika wraz z komunikatem trasy śledzenia, pod warunkiem, że ich aplikacje mogą sobie poradzić z poprzednimi parametrami PCF.

Nagłówek PCF **musi** być pierwszym nagłówkiem, albo komunikat nie będzie traktowany jako komunikat trasy śledzenia. Oznacza to, że komunikat nie może znajdować się w grupie, a komunikaty śledzenia trasy nie mogą być segmentowane. Jeśli komunikat trasy śledzenia zostanie wysłany w grupie, komunikat zostanie odrzucony z kodem przyczyny MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Należy pamiętać, że wartość MQFMT_ADMIN może być również używana dla formatu komunikatu trasy śledzenia, ale w tym przypadku nie można wysłać danych użytkownika wraz z komunikatem trasy śledzenia.

Zdarzenie MQFMT_EVENT

Komunikat jest komunikatem o zdarzeniu MQ , który raportuje zdarzenie, które wystąpiło. Komunikaty zdarzeń mają taką samą strukturę jak komendy programowalne; więcej informacji na temat tej struktury można znaleźć w sekcji [Komunikaty komend PCF](#) , a w sekcji [Monitorowanie zdarzeń](#) -informacje o zdarzeniach.

Komunikaty zdarzeń Version-1 mogą być przekształcane we wszystkich środowiskach, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT. Komunikaty zdarzeń Version-2 mogą być przekształcane tylko w z/OS.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_EVENT_ARRAY. Wartość ta ma taką samą wartość jak MQFMT_EVENT, ale jest tablicą znaków zamiast łańcucha.

MQFMT_IMS

Dane komunikatu rozpoczynają się od nagłówka informacyjnego IMS MQIIH, po którym następuje dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *Format* w strukturze MQIIH.

Szczegółowe informacje na temat sposobu obsługi struktury MQIIH podczas korzystania z komendy MQGET z opcją MQGMO_CONVERT można znaleźć w sekcji [“Format \(MQCHAR8\)”](#) na stronie 403 i [“Format ReplyTo\(MQCHAR8\)”](#) na stronie 403.

Dla języka programowania w języku C definiowana jest również stała MQFMT_IMS_ARRAY. Ma ona taką samą wartość co MQFMT_IMS, ale jest tablicą znaków zamiast łańcucha.

MQFMT_IMS_VAR_STRING

Komunikat jest łańcuchem zmiennej IMS, który jest łańcuchem w postaci 11zzccc, gdzie:

11

jest 2-bajtowym polem długości określaniem łącznej długości elementu łańcucha zmiennej IMS. Ta długość jest równa długości 11 (2 bajty), powiększonej o długość zz (2 bajty), plus długość łańcucha znaków. 11 to dwubajtowa binarna liczba całkowita w kodowaniu określonym w polu *Encoding*.

zz

to dwubajtowe pole zawierające flagi istotne dla IMS. zz jest łańcuchem bajtowym składającym się z dwóch pól MQBYTE i jest przesyłany bez zmiany nadawcy do odbiorcy (to znaczy, że zz nie podlega żadnej konwersji).

ccc

to łańcuch znaków o zmiennej długości zawierający znaki 11-4. ccc znajduje się w zestawie znaków określonym w polu *CodedCharSetId*.

W systemie z/OS dane komunikatu mogą składać się z sekwencji łańcuchów zmiennych IMS, które zostały połączone razem, przy czym każdy łańcuch ma postać 11zzccc. Między kolejnymi łańcuchami zmiennych IMS nie może być pominiętych żadnych bajtów. Oznacza to, że jeśli pierwszy łańcuch ma nieparzystą długość, drugi łańcuch będzie błędnie wyrównany, to znaczy, że nie rozpocznie się on na granicy, która jest wielokrotnością dwóch. Należy zachować ostrożność przy konstruowaniu takich łańcuchów na komputerach, które wymagają wyrównania elementarnych typów danych.

Użyj opcji MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić komunikaty, które mają format MQFMT_IMS_VAR_STRING.

Dla języka programowania C jest również zdefiniowana stała zmienna MQFMT_IMS_VAR_STRING_ARRAY. Ma ona taką samą wartość jak MQFMT_IMS_VAR_STRING, ale jest tablicą znaków zamiast łańcucha.

MQFMT_MD_EXTENSION

Dane komunikatu rozpoczynają się od rozszerzenia deskryptora komunikatu MQMDE, a następnie są śledzone innymi danymi (zwykle są to dane komunikatu aplikacji). Nazwa formatu, zestaw znaków i kodowanie danych, które są zgodne z MQMDE, są nadawane przez pola *Format*, *CodedCharSetId* i *Encoding* w produkcie MQMDE. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji [“MQMDE-Rozszerzenie deskryptora komunikatu”](#) na stronie 471. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_MD_EXTENSION_ARRAY. Ma ona taką samą wartość co MQFMT_MD_EXTENSION, ale jest tablicą znaków zamiast łańcucha.

MQFMT_PCF

Komunikat jest komunikatem definiowanym przez użytkownika, który jest zgodny ze strukturą komunikatu PCF (programmable command format). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT. Więcej informacji

na temat używania programowalnych komunikatów formatu komend zawiera sekcja Korzystanie z formatów komend programowalnych .

Dla języka programowania w języku C definiowana jest również stała MQFMT_PCF_ARRAY. Ma ona taką samą wartość jak MQFMT_PCF, ale jest tablicą znaków zamiast łańcucha.

MQFMT_REF_MSG_HEADER

Dane komunikatu rozpoczynają się od nagłówka komunikatu odwołania MQRMH i są opcjonalnie śledzone przez inne dane. Nazwa formatu, zestaw znaków i kodowanie danych są podawane za pomocą pól *Format*, *CodedCharSetId* i *Encoding* w tabeli MQRMH. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji “MQRMH-nagłówki komunikatu odwołania” na stronie 550 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Ten format jest obsługiwany w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_REF_MSG_HEADER_ARRAY; ma ona taką samą wartość jak MQFMT_REF_MSG_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_RF_HEADER

Dane komunikatu rozpoczynają się od reguły i nagłówka formatowania MQRFH, a następnie są śledzone innymi danymi. Nazwa formatu, zestaw znaków i kodowanie danych (jeśli istnieją) są nadawane przez pola *Format*, *CodedCharSetId* i *Encoding* w MQRFH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_RF_HEADER_ARRAY. Wartość ta ma taką samą wartość jak MQFMT_RF_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_RF_HEADER_2

Dane komunikatu rozpoczynają się od reguły version-2 i nagłówka formatowania MQRFH2, a następnie są śledzone innymi danymi. Nazwa formatu, zestaw znaków i kodowanie danych opcjonalnych (jeśli istnieją) są nadawane przez pola *Format*, *CodedCharSetId* i *Encoding* w tabeli MQRFH2. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

W przypadku języka programowania w języku C jest również zdefiniowana stała MQFMT_RF_HEADER_2_ARRAY, która ma taką samą wartość jak MQFMT_RF_HEADER_2, ale jest tablicą znaków zamiast łańcucha.

MQFMT_STRING

Dane komunikatu aplikacji mogą być łańcuchami SBCS (zestaw znaków jednobajtowych) lub łańcuchami DBCS (zestaw znaków dwubajtowych). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_STRING_ARRAY. Ma ona taką samą wartość jak MQFMT_STRING, ale jest tablicą znaków zamiast łańcucha.

MQFMT_TRIGGER

Komunikat jest komunikatem wyzwalanym, opisanym przez strukturę MQTM; szczegółowe informacje na temat tej struktury zawiera sekcja “MQTM-komunikat wyzwalacza” na stronie 603 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania w języku C jest również zdefiniowana stała tablica MQFMT_TRIGGER_ARRAY. Ma ona taką samą wartość jak MQFMT_TRIGGER, ale jest tablicą znaków zamiast łańcucha.

Nagłówek MQFMT_WORK_INFO_HEADER

Dane komunikatu rozpoczynają się od nagłówka informacji o pracy MQWIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *Format* w strukturze MQWIH.

W systemie z/OS należy określić opcję MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić dane użytkownika w komunikaty o formacie MQFMT_WORK_INFO_HEADER. Jednak sama struktura

MQWIIH jest zawsze zwracana w zestawie znaków i kodowaniu menedżera kolejek (to znaczy, że struktura MQWIIH jest przekształcana bez względu na to, czy opcja MQGMO_CONVERT jest określona).

Dla języka programowania C zdefiniowana jest również stała MQFMT_WORK_INFO_HEADER_ARRAY; ma ona taką samą wartość jak MQFMT_WORK_INFO_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_XMIT_Q_HEADER

Dane komunikatu rozpoczynają się od nagłówka kolejki transmisji MQXQH. Dane z oryginalnego komunikatu są natychmiast następujące po strukturze MQXQH. Nazwa formatu oryginalnych danych komunikatu jest podawana przez pole *Format* w strukturze MQMD, która jest częścią nagłówka kolejki transmisji MQXQH. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji [“MQXQH-nagłówki kolejki transmisji”](#) na stronie 623.

Raporty COA i COD nie są generowane dla komunikatów, które mają *Format* o wartości MQFMT_XMIT_Q_HEADER.

Dla języka programowania w języku C jest również zdefiniowana stała MQFMT_XMIT_Q_HEADER_ARRAY; ma ona taką samą wartość jak MQFMT_XMIT_Q_HEADER, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

GroupId (MQBYTE24)

Jest to łańcuch bajtowy używany do identyfikowania określonej grupy komunikatów lub komunikatu logicznego, do którego należy komunikat fizyczny. *GroupId* jest również używany, jeśli dla komunikatu dozwolona jest segmentacja. We wszystkich tych przypadkach wartość *GroupId* ma wartość inną niż NULL, a w polu *MsgFlags* ustawiona jest co najmniej jedna z następujących opcji:

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

Jeśli żadna z tych opcji nie jest ustawiona, program *GroupId* ma specjalną wartość NULL MQGI_NONE.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET wartość MQMO_MATCH_GROUP_ID *nie* jest określona.

Poniżej przedstawiono zalecane sposoby korzystania z tych wywołań w przypadku komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołania to MQPUT1, aplikacja musi upewnić się, że parametr *GroupId* jest ustawiony na odpowiednią wartość.

Grupy komunikatów i segmenty mogą być przetwarzane poprawnie tylko wtedy, gdy identyfikator grupy jest unikalny. Z tego powodu *aplikacje nie mogą generować własnych identyfikatorów grup*; zamiast tego aplikacje muszą wykonać jedną z następujących czynności:

- Jeśli określono parametr MQPMO_LOGICAL_ORDER, menedżer kolejek automatycznie generuje unikalny identyfikator grupy dla pierwszego komunikatu w grupie lub segmencie komunikatu logicznego i używa tego identyfikatora grupy dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego, dlatego aplikacja nie musi podejmować żadnych specjalnych działań. Jest to zalecana procedura.
- Jeśli parametr MQPMO_LOGICAL_ORDER *nie* jest określony, aplikacja musi zażądać, aby menedżer kolejek wygenerował identyfikator grupy, ustawiając parametr *GroupId* na wartość MQGI_NONE w pierwszej wywołaniu MQPUT lub MQPUT1 dla komunikatu w grupie lub segmencie komunikatu logicznego. Identyfikator grupy zwracany przez menedżera kolejek na wyjściu z tego wywołania musi

być następnie używany dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego. Jeśli grupa komunikatów zawiera segmentowane komunikaty, to ten sam identyfikator grupy musi być używany dla wszystkich segmentów i komunikatów w grupie.

Jeśli parametr MQPMO_LOGICAL_ORDER nie jest określony, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności (na przykład w kolejności odwrotnej), ale identyfikator grupy musi być przydzielony przez *pierwsze* wywołanie MQPUT lub MQPUT1, które jest wydawane dla dowolnego z tych komunikatów.

W przypadku danych wejściowych wywołania MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji *Kolejność fizyczna w kolejce*. W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem, jeśli obiekt otwarty jest pojedynczą kolejką, a nie listą dystrybucyjną, ale pozostawia ją niezmienioną, jeśli otwarty obiekt jest listą dystrybucyjną. W tym drugim przypadku, jeśli aplikacja musi znać wygenerowane identyfikatory grup, aplikacja musi udostępnić rekordy MQPMR zawierające pole *GroupId*.

W przypadku wejścia do wywołania MQGET menedżer kolejek używa wartości opisanej w sekcji *Tabela 46 na stronie 388*. W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Zdefiniowane są następujące wartości specjalne:

MQGI_NONE

Nie określono identyfikatora grupy.

Wartość jest binarna zero dla długości pola. Jest to wartość używana dla komunikatów, które nie znajdują się w grupach, nie są segmentami komunikatów logicznych i dla których segmentacja nie jest dozwolona.

W przypadku języka programowania C zdefiniowana jest również stała MQGI_NONE_ARRAY; ta wartość ma taką samą wartość jak MQGI_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_GROUP_ID_LENGTH. Wartością początkową tego pola jest MQGI_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MsgFlags (MQLONG)

MsgFlags to opcje, które określają atrybuty komunikatu lub sterują jego przetwarzaniem.

MsgFlags są podzielone na następujące kategorie:

- Flagi segmentacji
- Flagi statusu

Flagi segmentacji: jeśli komunikat jest zbyt duży dla kolejki, próba umieszczenia komunikatu w kolejce zwykle nie powiedzie się. Segmentacja to technika, za pomocą której menedżer kolejek lub aplikacja dzieli komunikat na mniejsze części zwane segmentami i umieszcza każdy segment w kolejce jako osobny komunikat fizyczny. Aplikacja, która pobiera komunikat, może pobrać segmenty jeden za pomocą jednego lub poprosić menedżera kolejek w celu ponownego złożenia segmentów w jeden komunikat zwracany przez wywołanie MQGET. Ten ostatni jest osiągnięty przez określenie opcji MQGMO_COMPLETE_MSG w wywołaniu MQGET i dostarczenie buforu, który jest wystarczająco duży, aby pomieścić pełny komunikat. (Szczegółowe informacje na temat opcji MQGMO_COMPLETE_MSG można znaleźć w sekcji “MQGMO-Opcje Get-message” na stronie 367). Komunikat może być segmentowany w wysyłającym menedżerze kolejek, w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek.

Aby sterować segmentacją komunikatu, można określić jedną z następujących opcji:

MQMF_SEGMENTATION_INHIBITED

Ta opcja uniemożliwia rozbięcie komunikatu na segmenty przez menedżer kolejek. Jeśli zostanie określona dla komunikatu, który jest już segmentem, ta opcja uniemożliwia rozbięcie segmentu na mniejsze segmenty.

Wartość tej flagi jest binarna zero. Jest to opcja domyślna.

MQMF_SEGMENTATION_ALLOWED

Ta opcja umożliwia rozbicie komunikatu na segmenty przez menedżer kolejek. Jeśli zostanie określona dla komunikatu, który jest już segmentem, ta opcja umożliwia rozbicie segmentu na mniejsze segmenty. Parametr MQMF_SEGMENTATION_ALLOWED może być ustawiony bez ustawionego parametru MQMF_SEGMENT lub MQMF_LAST_SEGMENT.

- W systemie z/OS menedżer kolejek nie obsługuje segmentacji komunikatów. Jeśli komunikat jest zbyt duży dla kolejki, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_MSG_TOO_BIG_FOR_Q. Można jednak nadal określić opcję MQMF_SEGMENTATION_ALLOWED, która umożliwia segmentację komunikatu w zdalnym menedżerze kolejek.

Gdy menedżer kolejek segmentuje komunikat, menedżer kolejek włącza flagę MQMF_SEGMENT w kopii deskryptora MQMD, który jest wysyłany z każdym segmentem, ale nie zmienia ustawień tych flag w strukturze MQMD udostępnianej przez aplikację w wywołaniu MQPUT lub MQPUT1. Dla ostatniego segmentu w komunikacie logicznym menedżer kolejek włącza również flagę MQMF_LAST_SEGMENT w strukturze MQMD, która jest wysyłana z segmentem.

Uwaga: Należy zachować ostrożność podczas umieszczania komunikatów z opcją MQMF_SEGMENTATION_ALLOWED, ale bez parametru MQPMO_LOGICAL_ORDER. Jeśli komunikat jest następujący:

- Nie jest to segment, oraz
- Nie w grupie, oraz
- Nie są przekazywane,

Aplikacja musi zresetować pole *GroupId* na wartość MQGI_NONE przed *każdym* wywołaniem MQPUT lub MQPUT1, tak aby menedżer kolejek mógł wygenerować unikalny identyfikator grupy dla każdego komunikatu. Jeśli nie jest to zrobione, niepowiązane komunikaty mogą mieć ten sam identyfikator grupy, co może prowadzić do niepoprawnego przetwarzania. Aby uzyskać więcej informacji na temat resetowania pola *GroupId*, należy zapoznać się z opisami pola *GroupId* oraz opcji MQPMO_LOGICAL_ORDER.

Menedżer kolejek rozdziela komunikaty do segmentów w razie potrzeby, tak aby segmenty (oraz wszystkie wymagane dane nagłówka) pasowały do kolejki. Istnieje jednak dolna granica wielkości segmentu generowanego przez menedżer kolejek, a tylko ostatni segment utworzony z komunikatu może być mniejszy niż ten limit (dolny limit dla wielkości segmentu wygenerowanego przez aplikację to jeden bajt). Segmenty wygenerowane przez menedżer kolejek mogą mieć nierówną długość. Menedżer kolejek przetwarza komunikat w następujący sposób:

- Formaty zdefiniowane przez użytkownika są podzielone na granice, które są wielokrotnością 16 bajtów; menedżer kolejek nie generuje segmentów o wielkości mniejszej niż 16 bajtów (innych niż ostatni segment).
- Wbudowane formaty inne niż MQFMT_STRING są dzielone w punktach odpowiednich do charakteru prezentowanych danych. Jednak menedżer kolejek nigdy nie splituje komunikatu w środku struktury nagłówka IBM MQ. Oznacza to, że segment zawierający pojedynczą strukturę nagłówka MQ nie może być dalej dzielony przez menedżer kolejek, a w rezultacie minimalny możliwy rozmiar segmentu dla tego komunikatu jest większy niż 16 bajtów.

Drugi lub późniejszy segment wygenerowany przez menedżer kolejek rozpoczyna się od jednego z następujących elementów:

- Struktura nagłówka MQ
- Początek danych komunikatu aplikacji
- Część drogi za pośrednictwem danych komunikatu aplikacji
- MQFMT_STRING jest dzielony bez względu na rodzaj prezentowanych danych (SBCS, DBCS lub mieszane SBCS/DBCS). Jeśli tańcuch jest typu DBCS lub mieszany SBCS/DBCS, może to spowodować, że segmenty, których nie można przekształcić z jednego zestawu znaków na inny, mogą być przekształcane. Menedżer kolejek nigdy nie splituje komunikatów MQFMT_STRING w segmenty o wielkości mniejszej niż 16 bajtów (inne niż ostatni segment).

- Menedżer kolejek ustawia pola *Format*, *CodedCharSetIdi Encoding* w strukturze MQMD każdego segmentu w celu poprawnego opisanie danych znajdujących się w *starcie* segmentu. Nazwa formatu to nazwa wbudowanego formatu lub nazwa formatu zdefiniowanego przez użytkownika.
- Modyfikowana jest zmienna *Report* w strukturze MQMD segmentów o wartości *Offset* większej niż zero. Dla każdego typu raportu, jeśli opcja raportu to MQRO_*_WITH_DATA, ale segment nie może zawierać żadnego z pierwszych 100 bajtów danych użytkownika (to znaczy danych następujących po strukturach nagłówek IBM MQ, które mogą być obecne), opcja raportu zostanie zmieniona na MQRO_*.

Menedżer kolejek jest zgodny z powyższymi regułami, ale w przeciwnym razie komunikaty podziału są nieprzewidywalne. Nie należy wprowadzać założeń dotyczących miejsca, w którym komunikat jest podzielony.

W przypadku komunikatów *trwałych* menedżer kolejek może wykonywać segmentację tylko w ramach jednostki pracy:

- Jeśli wywołanie MQPUT lub MQPUT1 działa w ramach jednostki pracy zdefiniowanej przez użytkownika, ta jednostka pracy jest używana. Jeśli wywołanie nie powiedzie się w trakcie procesu segmentacji, menedżer kolejek usunie wszystkie segmenty, które zostały umieszczone w kolejce w wyniku niesprawnego wywołania. Jednak niepowodzenie nie uniemożliwia pomyślnego wykonania jednostki pracy.
- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, a nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie zakończy się pomyślnie, menedżer kolejek zatwierdza jednostkę pracy automatycznie. Jeśli wywołanie nie powiedzie się, menedżer kolejek wytworzy kopię zapasową jednostki pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika, ale istnieje zdefiniowana przez użytkownika jednostka pracy, menedżer kolejek nie może wykonać segmentacji. Jeśli komunikat nie wymaga segmentacji, wywołanie może zakończyć się powodzeniem. Jeśli jednak komunikat wymaga segmentacji, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_UOW_NOT_AVAILABLE.

W przypadku komunikatów *nietrwałych* menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu przeprowadzenia segmentacji.

Podczas przekształcania danych w komunikaty, które mogą być posegmentowane, należy zachować szczególną ostrożność:

- Jeśli aplikacja odbierający przekształca dane w wywołaniu MQGET i określa opcję MQGMO_COMPLETE_MSG, wyjście konwersji danych jest przekazywane kompletnym komunikatem dla wyjścia do przekształcenia, a fakt, że komunikat był segmentowany, jest widoczny dla wyjścia.
- Jeśli aplikacja odbierający pobiera jeden segment jednocześnie, to wyjście konwersji danych jest wywoływane w celu przekształcenia jednego segmentu w danym momencie. W związku z tym wyjście musi przekształcić dane w segment niezależnie od danych w dowolnym z pozostałych segmentów.

Jeśli rodzaj danych w komunikacie jest taki, że arbitralna segmentacja danych na granicy 16-bajtowej może spowodować, że segmenty nie mogą zostać przekształcone przez wyjście, lub jeśli formatem jest MQFMT_STRING, a zestaw znaków to DBCS lub mieszane SBCS/DBCS, aplikacja wysyłający musi utworzyć i umieścić segmenty, określając wartość MQMF_SEGMENTATION_INHIBITED w celu zablokowania dalszej segmentacji. W ten sposób aplikacja wysyłający może zapewnić, że każdy segment będzie zawierał wystarczającą ilość informacji, aby umożliwić wyjście konwersji danych w celu pomyślnego przekształcenia segmentu.

- Jeśli dla wysyłającego agenta kanału komunikatów (MCA) określono konwersję nadawcy, agent MCA przekształca tylko komunikaty, które nie są segmentami komunikatów logicznych; agent MCA nigdy nie próbuje konwertować komunikatów, które są segmentami.

Ta opcja jest flagą wejściową w wywołaniach MQPUT i MQPUT1 oraz flagą wyjściową wywołania MQGET. W przypadku tego ostatniego wywołania menedżer kolejek również odbija wartość flagi w polu *Segmentation* w produkcie MQGMO.

Wartością początkową tej flagi jest MQMF_SEGMENTATION_INHIBITED.

flagi statusu: są to flagi, które wskazują, czy komunikat fizyczny należy do grupy komunikatów, czy jest to segment komunikatu logicznego, czy też żaden z nich. W wywołaniu MQPUT lub MQPUT1 lub zwróconej przez wywołanie MQGET można określić co najmniej jedną z następujących wartości:

MQMF_MSG_IN_GROUP

Wiadomość jest członkiem grupy.

MQMF_LAST_MSG_IN_GROUP

Komunikat jest ostatnim komunikatem logicznym w grupie.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza MQMF_MSG_IN_GROUP w kopii deskryptora MQMD, który jest wysyłany z komunikatem, ale nie zmienia ustawień tych flag w strukturze MQMD udostępnianej przez aplikację w wywołaniu MQPUT lub MQPUT1.

Grupa może składać się tylko z jednego komunikatu logicznego. W takim przypadku ustawiona jest wartość MQMF_LAST_MSG_IN_GROUP, ale wartość w polu *MsgSeqNumber* jest ustawiona na wartość 1.

MQMF_SEGMENT

Komunikat jest segmentem komunikatu logicznego.

Jeśli wartość MQMF_SEGMENT została określona bez parametru MQMF_LAST_SEGMENT, długość danych komunikatu aplikacji w segmencie (*wykluczanie* długości wszystkich struktur nagłówka IBM MQ , które mogą być obecne) musi być co najmniej jedna. Jeśli długość wynosi zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_SEGMENT_LENGTH_ZERO.

W systemie z/OSA opcja nie jest obsługiwana, jeśli komunikat jest umieszczany w kolejce o typie indeksu MQIT_GROUP_ID.

MQMF_LAST_SEGMENT

Komunikat jest ostatnim segmentem komunikatu logicznego.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza program MQMF_SEGMENT w kopii deskryptora MQMD, który jest wysyłany z komunikatem, ale nie zmienia ustawień tych flag w deskryptywie MQMD udostępnionym przez aplikację w wywołaniu MQPUT lub MQPUT1.

Komunikat logiczny może składać się tylko z jednego segmentu. Jeśli tak, to parametr MQMF_LAST_SEGMENT jest ustawiony, ale pole *Offset* ma wartość zero.

Jeśli określono wartość MQMF_LAST_SEGMENT, długość danych komunikatu aplikacji w segmencie (*wykluczanie* długości wszystkich struktur nagłówka, które mogą być obecne) może wynosić zero.

W systemie z/OSA opcja nie jest obsługiwana, jeśli komunikat jest umieszczany w kolejce o typie indeksu MQIT_GROUP_ID.

Podczas umieszczania komunikatów aplikacja musi upewnić się, że flagi są poprawnie ustawione. Jeśli określono parametr MQPMO_LOGICAL_ORDER lub został określony w poprzedzającym wywołaniu MQPUT dla uchwytu kolejki, ustawienia flag muszą być spójne z informacjami o grupach i segmentach zachowywanych przez menedżer kolejek dla uchwytu kolejki. Następujące warunki mają zastosowanie do *kolejnych* wywołań MQPUT dla uchwytu kolejki, jeśli określono parametr MQPMO_LOGICAL_ORDER:

- Jeśli nie ma żadnej bieżącej grupy lub komunikatu logicznego, wszystkie te opcje (i ich kombinacje) są poprawne.
- Jeśli określono parametr MQMF_MSG_IN_GROUP, musi on pozostać w miejscu, dopóki nie zostanie podana wartość MQMF_LAST_MSG_IN_GROUP. Wywołanie nie powiodło się z kodem przyczyny MQRC_INCOMPLETE_GROUP, jeśli ten warunek nie jest spełniony.
- Jeśli określono parametr MQMF_SEGMENT, musi on pozostać w określonym czasie, dopóki nie zostanie podana wartość MQMF_LAST_SEGMENT. Wywołanie nie powiodło się z kodem przyczyny MQRC_INCOMPLETE_MSG, jeśli ten warunek nie jest spełniony.
- Po określeniu parametru MQMF_SEGMENT bez MQMF_MSG_IN_GROUP, parametr MQMF_MSG_IN_GROUP musi pozostać *off*, dopóki nie zostanie podana wartość parametru

MQMF_LAST_SEGMENT. Wywołanie nie powiodło się z kodem przyczyny MQRC_INCOMPLETE_MSG, jeśli ten warunek nie jest spełniony.

W polu Kolejność fizyczna w kolejce wyświetlane są poprawne kombinacje flag oraz wartości używane dla różnych pól.

Opcje te są flagami wejściowymi w wywołaniach MQPUT i MQPUT1, a także flagi wyjściowe w wywołaniu MQGET. W tym drugim wywołaniu menedżer kolejek również odbija wartości flag dla pól *GroupStatus* i *SegmentStatus* w produkcie MQGMO.

Nie można używać zgrupowanych ani segmentowanych komunikatów z publikowania/subskrybowania.

Opcje domyślne: można określić, że komunikat ma atrybuty domyślne:

MQMF_NONE

Brak flag komunikatów (domyślne atrybuty komunikatu).

Powoduje to zahamowanie segmentacji i wskazuje, że komunikat nie znajduje się w grupie i nie jest segmentem komunikatu logicznego. Wartość MQMF_NONE jest definiowana w celu uzyskania dokumentacji programu pomocowego. Nie jest zamierzone, aby ta opcja była używana z innymi, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Pole *MsgFlags* jest partycjonowane w podpola; szczegółowe informacje na ten temat zawiera sekcja “Opcje raportów i flagi komunikatów” na stronie 915.

Wartością początkową tego pola jest MQMF_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MsgId (MQBYTE24)

Jest to łańcuch bajtowy używany do odróżniania jednego komunikatu od innego. Ogólnie, dwa komunikaty nie powinny mieć tego samego identyfikatora komunikatu, chociaż nie jest to niedozwolone przez menedżer kolejek. Identyfikator komunikatu jest stałą właściwością komunikatu i utrzymuje się między restartami menedżera kolejek. Ponieważ identyfikator komunikatu jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator komunikatu *nie* jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do drugiego.

W przypadku wywołań MQPUT i MQPUT1, jeśli aplikacja MQMI_NONE lub MQPMO_NEW_MSG_ID jest określona przez aplikację, menedżer kolejek generuje unikalny identyfikator komunikatu.²po umieszczeniu komunikatu i umieszcza go w deskrypcorze komunikatu wysłanym razem z komunikatem. Menedżer kolejek zwraca również ten identyfikator komunikatu w deskrypcorze komunikatu należącym do aplikacji wysyłającej. Aplikacja ta może używać tej wartości do rejestrowania informacji o konkretnych komunikatach, a także do odpowiadania na zapytania z innych części aplikacji.

Jeśli komunikat jest umieszczany w temacie, menedżer kolejek generuje unikalne identyfikatory komunikatów, jeśli jest to konieczne dla każdego opublikowanego komunikatu. Jeśli aplikacja MQPMO_NEW_MSG_ID jest określona przez aplikację, menedżer kolejek generuje unikalny identyfikator komunikatu w celu zwrócenia danych wyjściowych. Jeśli aplikacja MQMI_NONE jest określona przez aplikację, wartość pola *MsgId* w strukturze MQMD nie zmienia się po powrocie z wywołania.

Więcej informacji na temat zachowanych publikacji znajduje się w opisie komendy MQPMO_RETAIN w produkcie “Opcje MQPMO (MQLONG)” na stronie 507.

² *MsgId* wygenerowany przez menedżer kolejek składa się z 4-bajтового identyfikatora produktu (AMQ → lub CSQ → w ASCII lub EBCDIC, gdzie → oznacza pusty znak), po którym następuje implementacja specyficzna dla produktu w postaci unikalnego łańcucha. W produkcie IBM MQ jest to pierwsze 12 znaków nazwy menedżera kolejek oraz wartość uzyskana z zegara systemowego. Dlatego wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się od pierwszych 12 znaków, aby identyfikatory komunikatów były unikalne. Możliwość wygenerowania unikalnego łańcucha zależy również od tego, że zegar systemowy nie jest zmieniany wstecz. Aby wyeliminować możliwość utworzenia identyfikatora komunikatu wygenerowanego przez menedżer kolejek duplikujący wygenerowany przez aplikację, aplikacja musi unikać generowania identyfikatorów z początkowymi znakami z zakresu od A do I w kodzie ASCII lub EBCDIC (X'41 'do X'49' i X'C1'do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, menedżer kolejek generuje unikalne identyfikatory komunikatów w razie potrzeby, ale wartość pola *MsgId* w strukturze MQMD jest niezmieniona po powrocie z wywołania, nawet jeśli określono wartość MQMI_NONE lub MQPMO_NEW_MSG_ID. Jeśli aplikacja musi znać identyfikatory komunikatów wygenerowane przez menedżer kolejek, aplikacja musi udostępnić rekordy MQPMR zawierające pole *MsgId*.

Aplikacja wysyłający może również określić wartość dla identyfikatora komunikatu innego niż MQMI_NONE; spowoduje to zatrzymanie menedżera kolejek generującego unikalny identyfikator komunikatu. Aplikacja, która przekazuje komunikat, może użyć tej aplikacji do propagowania identyfikatora komunikatu oryginalnego.

Menedżer kolejek nie używa tego pola z wyjątkiem:

- Generuj unikalną wartość, jeśli jest to wymagane, zgodnie z opisem powyżej
- Dostarcz wartość do aplikacji, która wydaje żądanie pobrania dla komunikatu.
- Skopiuj wartość do pola *CorrelId* dowolnego komunikatu raportu generowanego na temat tego komunikatu (w zależności od opcji *Report*).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole *MsgId* w sposób określony przez pole *Report* oryginalnego komunikatu, MQRO_NEW_MSG_ID lub MQRO_PASS_MSG_ID. Aplikacje, które generują komunikaty raportów, muszą również to zrobić.

W przypadku wywołania MQGET *MsgId* jest jednym z pięciu pól, które mogą być używane do pobierania konkretnego komunikatu z kolejki. Zwykle wywołanie MQGET zwraca następny komunikat w kolejce, ale konkretny komunikat można uzyskać, podając co najmniej jedno z pięciu kryteriów wyboru, w dowolnej kombinacji. Te pola są następujące:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

Aplikacja ustawia jedno lub więcej z tych pól na wymagane wartości, a następnie ustawia odpowiednie opcje zgodności MQMO_* w polu *MatchOptions* w produkcie MQGMO w celu użycia tych pól jako kryteriów wyboru. Tylko komunikaty, które mają określone wartości w tych polach, są kandydatami do pobrania. Wartość domyślna dla pola *MatchOptions* (jeśli nie została zmieniona przez aplikację) jest zgodna zarówno z identyfikatorem komunikatu, jak i identyfikatorem korelacji.

W systemie z/OS kryteria wyboru, których można użyć, są ograniczone przez typ indeksu używanego w kolejce. Szczegółowe informacje znajdują się w atrybucie kolejki *IndexType*.

Zwykle zwracany jest komunikat *pierwszy* w kolejce, który spełnia kryteria wyboru. Jeśli jednak określono parametr MQGMO_BROWSE_NEXT, zwracany jest komunikat *next*, który spełnia kryteria wyboru. Skanowanie dla tego komunikatu rozpoczyna się od komunikatu *następującego*, w którym znajduje się bieżąca pozycja kursora.

Uwaga: Kolejka jest skanowana sekwencyjnie w przypadku komunikatu spełniającego kryteria wyboru, dlatego czasy pobierania są wolniejsze niż w przypadku, gdy nie określono kryteriów wyboru, szczególnie jeśli przed znalezieniem odpowiedniej liczby komunikatów musi być skanowany wiele komunikatów. Wyjątki od tego są następujące:

- wywołanie MQGET przez parametr *CorrelId* na 64-bitowych platformach rozproszonych, w których indeks *CorrelId* eliminuje konieczność wykonania prawdziwie sekwencyjnego skanowania.
- Wywołanie MQGET przez *IndexType* w systemie z/OS.

W obu tych przypadkach wydajność pobierania została poprawiona.

Więcej informacji na temat sposobu użycia kryteriów wyboru w różnych sytuacjach zawiera sekcja [Tabela 46 na stronie 388](#).

Określenie parametru MQMI_NONE jako identyfikatora komunikatu ma taki sam efekt, jak *nie* określenie identyfikatora komunikatu MQMO_MATCH_MSG_ID, czyli *any* jest zgodny z identyfikatorem komunikatu.

To pole jest ignorowane, jeśli opcja MQGMO_MSG_UNDER_CURSOR jest określona w parametrze *GetMsgOpts* w wywołaniu MQGET.

W przypadku powrotu z wywołania MQGET pole *MsgId* jest ustawione na identyfikator komunikatu zwróconego przez komunikat (jeśli istnieje).

Można użyć następującej wartości specjalnej:

MQMI_NONE

Nie określono identyfikatora komunikatu.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQMI_NONE_ARRAY; ta wartość ma taką samą wartość jak MQMI_NONE, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe/wyjściowe dla wywołań MQGET, MQPUT i MQPUT1. Długość tego pola jest podana przez wartość MQ_MSG_ID_LENGTH. Wartością początkową tego pola jest MQMI_NONE.

Liczba MsgSeq(MQLONG)

Jest to numer kolejny komunikatu logicznego w grupie.

Numery kolejne rozpoczynają się od 1, a następnie zwiększają się o 1 dla każdego nowego komunikatu logicznego w grupie, do 999 999 999. Numerem kolejnym komunikatu fizycznego będącego poza grupą jest 1.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET wartość MQMO_MATCH_MSG_SEQ_NUMBER jest *nie* określona.

Poniżej przedstawiono zalecane sposoby korzystania z tych wywołań w przypadku komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołania to MQPUT1, aplikacja musi upewnić się, że parametr *MsgSeqNumber* jest ustawiony na odpowiednią wartość.

W przypadku danych wejściowych wywołania MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji Kolejność fizyczna w kolejce. W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem.

W przypadku danych wejściowych wywołania MQGET menedżer kolejek używa wartości przedstawionej w produkcie Tabela 46 na stronie 388. W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Wartość początkowa tego pola jest jedną z wartości. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MsgType (MQLONG)

Wskazuje typ komunikatu. Typy komunikatów są pogrupowane w następujący sposób:

MQMT_SYSTEM_FIRST

Najniższa wartość dla typów komunikatów zdefiniowanych przez system.

MQMT_SYSTEM_LAST

Najwyższa wartość dla typów komunikatów zdefiniowanych przez system.

Obecnie w zakresie systemu są zdefiniowane następujące wartości:

MQMT_DATAGRAM

Komunikat jest taki, że nie wymaga odpowiedzi.

MQMT_REQUEST

Komunikat jest taki, że wymaga odpowiedzi.

Podaj nazwę kolejki, do której ma zostać wysłana odpowiedź w polu *ReplyToQ*. Pole *Report* wskazuje, w jaki sposób należy ustawić *MsgId* i *CorrelId* odpowiedzi.

MQMT_REPLY

Komunikat jest odpowiedzią na wcześniejszy komunikat żądania (MQMT_REQUEST). Komunikat musi zostać wysłany do kolejki wskazanej w polu *ReplyToQ* komunikatu żądania. Użyj pola *Report* w żądaniu, aby określić sposób ustawienia *MsgId* i *CorrelId* odpowiedzi.

Uwaga: Menedżer kolejek nie wymusza relacji żądanie-odpowiedź. Jest to odpowiedzialność za aplikację.

Raport_menedżera_mQMT

Komunikat zgłasza oczekiwane lub nieoczekiwane zdarzenie, zwykle związane z jakimś innym komunikatem (na przykład odebrano komunikat żądania, który zawierał niepoprawne dane). Wyślij komunikat do kolejki wskazanej w polu *ReplyToQ* deskryptora komunikatu oryginalnego komunikatu. Ustaw pole *Feedback*, aby wskazać rodzaj raportu. Użyj pola *Report* oryginalnego komunikatu, aby określić sposób ustawienia *MsgId* i *CorrelId* komunikatu raportu.

Komunikaty raportów wygenerowane przez menedżera kolejek lub agenta kanału komunikatów są zawsze wysyłane do kolejki produktu *ReplyToQ*, a pola *Feedback* i *CorrelId* są ustawione zgodnie z powyższym opisem.

Możliwe jest również użycie wartości zdefiniowanych przez aplikację. Muszą one znajdować się w następującym zakresie:

MQMT_APPL_FIRST

Najniższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

MQMT_APPL_LAST

Najwyższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

W przypadku wywołań MQPUT i MQPUT1 wartość *MsgType* musi mieścić się w zakresie zdefiniowanym przez system albo w zakresie zdefiniowanym przez aplikację. Jeśli tak nie jest, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MSG_TYPE_ERROR.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Początkowa wartość tego pola to MQMT_DATAGRAM.

Przesunięcie (MQLONG)

Jest to przesunięcie w bajtach danych w komunikacie fizycznym od początku komunikatu logicznego, którego część stanowi część danych. Dane te nazywane są *segmentem*. Przesunięcie mieści się w zakresie od 0 do 999 999 999. Komunikat fizyczny, który nie jest segmentem komunikatu logicznego, ma przesunięcie zerowe.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET wartość MQMO_MATCH_OFFSET *nie* jest określona.

Poniżej przedstawiono zalecane sposoby korzystania z tych wywołań w przypadku komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja nie spełnia tych warunków lub wywołanie to MQPUT1, aplikacja musi upewnić się, że parametr *Offset* jest ustawiony na odpowiednią wartość.

W przypadku danych wejściowych wywołania MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji [Kolejność fizyczna w kolejce](#). W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem.

W przypadku raportu raportu dotyczącego segmentu komunikatu logicznego pole *OriginalLength* (pod warunkiem, że nie jest to MQOL_UNDEFINED) jest używane do aktualizowania przesunięcia w informacjach o segmencie zatrzymanych przez menedżer kolejek.

W przypadku danych wejściowych wywołania MQGET menedżer kolejek używa wartości przedstawionej w produkcie [Tabela 46 na stronie 388](#). W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Początkowa wartość tego pola wynosi zero. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

OriginalLength (MQLONG)

To pole ma znaczenie tylko w przypadku komunikatów raportu, które są segmentami. Określa długość segmentu wiadomości, do którego odnosi się komunikat raportu; nie określa długości komunikatu logicznego, którego część stanowi segment, lub długość danych w komunikacie raportu.

Uwaga: Podczas generowania komunikatu raportu dla komunikatu, który jest segmentem, menedżer kolejek i agent kanału komunikatów są kopiowane do deskryptora MQMD dla komunikatu raportu *GroupId, MsgSeqNumber, Offset, MsgFlags*, a pola z pierwotnego komunikatu. W związku z tym komunikat raportu jest również segmentem. Aplikacje, które generują komunikaty raportów, muszą działać w ten sam sposób i poprawnie ustawić pole *OriginalLength*.

Zdefiniowane są następujące wartości specjalne:

MQOL_NIEZDEFINIOWANY

Oryginalna długość komunikatu nie została zdefiniowana.

OriginalLength jest polem wejściowym w wywołaniach MQPUT i MQPUT1, ale wartość, którą udostępnia aplikacja, jest akceptowana tylko w określonych okolicznościach:

- Jeśli wstawiany komunikat jest segmentem i jest również komunikatem raportu, menedżer kolejek akceptuje podaną wartość. Wartość musi być następująca:

- Wartość większa od zera, jeśli segment nie jest ostatnim segmentem
- Wartość nie mniejsza niż zero, jeśli segment jest ostatnim segmentem.
- Nie mniej niż długość danych znajdujących się w komunikacie

Jeśli te warunki nie są spełnione, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ORIGINAL_LENGTH_ERROR.

- Jeśli wysyłany komunikat jest segmentem, ale nie jest komunikatem raportu, menedżer kolejek zignoruje pole i użyje zamiast niej długości danych komunikatu aplikacji.
- We wszystkich innych przypadkach menedżer kolejek ignoruje pole i zamiast niego korzysta z wartości MQOL_UNDEFINED.

To jest pole wyjściowe w wywołaniu MQGET.

Wartością początkową tego pola jest MQOL_UNDEFINED. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

Trwałość (MQLONG)

Wskazuje, czy komunikat jest zachowany po awariach systemu i restartach menedżera kolejek. W przypadku wywołań MQPUT i MQPUT1 wartość musi być jedną z następujących wartości:

MQPER_PERSISTENT

Komunikat przeżyje awarie systemu i restarty menedżera kolejek. Po umieszczeniu w komunikacie oraz jednostce pracy, w której został on umieszczony, został zatwierdzony (jeśli komunikat jest umieszczany jako część jednostki pracy), komunikat jest zachowany w pamięci dyskowej. Pozostaje tam do momentu usunięcia komunikatu z kolejki, a jednostka pracy, w której został on wczytany, została zatwierdzona (jeśli komunikat jest pobierany jako część jednostki pracy).

Gdy do kolejki zdalnej wysyłany jest komunikat trwały, wówczas mechanizm przechowywania i przekazywania przechowuje komunikat w każdym menedżerze kolejek wzdłuż trasy do miejsca docelowego, aż do momentu, gdy wiadomo, że komunikat dotarł do następnego menedżera kolejek.

Komunikaty trwałe nie mogą być umieszczane w następujących systemach:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, które są odwzorowywać na obiekt CFSTRUCT na poziomie CFLEVEL (2) lub poniżej, lub gdzie obiekt CFSTRUCT jest zdefiniowany jako RECOVER (NO).

Komunikaty trwałe mogą być umieszczane w statycznych kolejkach dynamicznych i predefiniowanych kolejkach.

MQPER_NOT_PERSISTENT

Komunikat nie jest zwykle restartowany w przypadku awarii systemu lub menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartowania menedżera kolejek w pamięci dyskowej znajduje się nienaruszona kopia komunikatu.

W przypadku nietrwałych komunikatów NPMCLASS (HIGH) komunikaty nietrwałe mogą przetrwać normalne zamknięcie i ponowne uruchomienie menedżera kolejek.

W przypadku kolejek współużytkowanych komunikaty nietrwałe przetrwają restarty menedżera kolejek w grupie współużytkowania kolejki, ale nie przeżywają niepowodzeń narzędzia CF używanego do przechowywania komunikatów w kolejkach współużytkowanych.

MQPER_PERSISTENCE_AS_Q_DEF

- Jeśli kolejka jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu *DefPersistence* zdefiniowanego w menedżerze kolejek *destination*, który jest właścicielem określonej instancji kolejki, w której umieszczony jest komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu *DefPersistence*, chociaż nie jest to wymagane.

Wartość *DefPersistence* jest kopiowana do pola *Persistence*, gdy komunikat jest umieszczany w kolejce docelowej. Jeśli później zostanie zmieniona opcja *DefPersistence*, komunikaty, które zostały już umieszczone w kolejce, nie będą miały wpływu na te komunikaty.

- Jeśli kolejka nie jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu *DefPersistence* zdefiniowanego w menedżerze kolejek *local*, nawet jeśli docelowy menedżer kolejek jest zdalny.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w definicji *pierwszej* w ścieżce. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Wartość *DefPersistence* jest kopiowana do pola *Persistence* po umieszczeniu w nim komunikacie. Jeśli później zostanie zmieniona wartość *DefPersistence*, komunikaty, które zostały już wprowadzone, nie zostaną naruszone.

Zarówno komunikaty trwałe, jak i nietrwałe mogą istnieć w tej samej kolejce.

Podczas odpowiadania na komunikat aplikacje muszą korzystać z trwałości komunikatu żądania dla komunikatu odpowiedzi.

W przypadku wywołania MQGET zwrócona wartość to MQPER_PERSISTENT lub MQPER_NOT_PERSISTENT.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Początkowa wartość tego pola to MQPER_PERSISTENCE_AS_Q_DEF.

Priorytet (MQLONG)

W przypadku wywołań MQPUT i MQPUT1 wartość ta musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następującej wartości specjalnej:

MQPRI_PRIORITY_AS_Q_DEF

- Jeśli kolejka jest kolejką klastra, priorytet komunikatu jest przyjmowany z atrybutu *DefPriority* zdefiniowanego w menedżerze kolejek *destination*, który jest właścicielem określonej instancji kolejki, w której umieszczony jest komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu *DefPriority*, chociaż nie jest to wymagane.

Wartość *DefPriority* jest kopiowana do pola *Priority* , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli później zostanie zmieniona opcja *DefPriority* , komunikaty, które zostały już umieszczone w kolejce, nie będą miały wpływu na te komunikaty.

- Jeśli kolejka nie jest kolejką klastra, priorytet komunikatu jest przyjmowany z atrybutu *DefPriority* zgodnie z definicją w *lokalnym* menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, priorytet domyślny jest przyjmowany z wartości tego atrybutu w definicji *pierwszej* w ścieżce. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Wartość *DefPriority* jest kopiowana do pola *Priority* po umieszczeniu w nim komunikacie. Jeśli później zostanie zmieniona wartość *DefPriority* , komunikaty, które zostały już wprowadzone, nie zostaną naruszone.

Wartość zwracana przez wywołanie MQGET jest zawsze większa lub równa zero; wartość MQPRI_PRIORITY_AS_Q_DEF nigdy nie jest zwracana.

Jeśli komunikat jest umieszczany z priorytetem większą niż maksymalna obsługiwana przez lokalny menedżer kolejek (wartość maksymalna jest podawana przez atrybut menedżera kolejek produktu *MaxPriority*), komunikat jest akceptowany przez menedżer kolejek, ale umieszczony w kolejce w maksymalnym priorytecie menedżera kolejek. Wywołanie MQPUT lub MQPUT1 kończy się z wartością MQCC_WARNING i kodem przyczyny MQRC_PRIORITY_PRZEKROCZENIA. Jednak pole *Priority* zachowuje wartość określoną przez aplikację, która wstawiła komunikat.

W systemie z/OS, jeśli komunikat o numerze MsgSeqo numerze 1 zostanie umieszczony w kolejce z sekwencją dostarczania komunikatów MQMDS_PRIORITY i typem indeksu MQIT_GROUP_ID, to kolejka może traktować komunikat o innym priorytecie. Jeśli komunikat został umieszczony w kolejce z priorytetem 0 lub 1, jest przetwarzany tak, jakby miał priorytet 2. Dzieje się tak dlatego, że kolejność komunikatów umieszczonych w tym typie kolejki jest zoptymalizowana w celu umożliwienia wydajnych testów kompletności grupy. Więcej informacji na temat sekwencji dostarczania komunikatów MQMDS_PRIORITY i typu indeksu MQIT_GROUP_ID zawiera sekcja [MsgDelivery](#), atrybut sekwencji.

Odpowiadając na komunikat, aplikacje muszą używać priorytetu komunikatu żądania dla komunikatu odpowiedzi. W innych sytuacjach podanie wartości MQPRI_PRIORITY_AS_Q_DEF umożliwia strojenie priorytetów bez zmiany aplikacji.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Początkowa wartość tego pola to MQPRI_PRIORITY_AS_Q_DEF.

Nazwa PutAppl(MQCHAR28)

Jest to nazwa aplikacji umieszczonej w komunikacie, która jest częścią *kontekstu źródłowego* komunikatu. Zawartość ta różni się między platformami i może się różnić od wersji.

Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i Kontekst komunikatu.

Format *PutApplName* zależy od wartości *PutApplType* i może być zmieniany z jednego wydania na inny. Zmiany są rzadkie, ale zdarzają się, jeśli zmiany w środowisku są wprowadzane.

Gdy menedżer kolejek ustawia to pole (czyli dla wszystkich opcji z wyjątkiem MQPMO_SET_ALL_CONTEXT), ustawia to pole na wartość, która jest określana przez środowisko:

-  W systemie z/OS menedżer kolejek używa:
 - Dla zadania wsadowego z/OS : 8-znakowa nazwa zadania z karty JES JOB.
 - Dla TSO, 7-znakowy identyfikator użytkownika TSO

- W przypadku systemu CICS: 8-znakowy identyfikator applid, po którym następuje 4-znakowy tranid.
- W przypadku systemu IMS: 8-znakowy identyfikator systemu IMS , po którym następuje 8-znakowa nazwa PSB.
- W przypadku XCF: 8-znakowa nazwa grupy XCF, po której następuje 16-znakowa nazwa elementu XCF
- W przypadku komunikatu wygenerowanego przez menedżer kolejek pierwsze 28 znaków nazwy menedżera kolejek
- W przypadku rozproszonego kolejkowania bez CICS, 8-znakowa nazwa zadania inicjatora kanału, po której następuje 8-znakowa nazwa modułu umieszczająca w kolejce niedostarczonych komunikatów, po której następuje 8-znakowy identyfikator zadania.

Nazwy lub nazwy są dopełniane spacjami z prawej strony, tak jak to jest w pozostałej części pola. W przypadku, gdy istnieje więcej niż jedna nazwa, nie ma między nimi separatora.

- **Windows** W systemach Windows menedżer kolejek używa następujących nazw:

- W przypadku aplikacji CICS nazwa transakcji CICS
- W przypadku aplikacji innej niż CICS należy podać co najwyżej 28 znaków pełnej nazwy pliku wykonywalnego.

- **IBM i** W systemie IBM imenedżer kolejek korzysta z pełnej nazwy zadania.

- **UNIX** W systemie Systemy UNIX menedżer kolejek używa następujących nazw:

- W przypadku aplikacji CICS nazwa transakcji CICS
- W przypadku aplikacji innej niż CICS produkt MQ zwraca się do systemu operacyjnego o nazwę procesu. Nazwa ta jest zwracana jako nazwa pliku programu bez pełnej ścieżki. Następnie program MQ umieszcza tę nazwę procesu w tabeli MQMD.PutApplName w następujący sposób:

AIX

Jeśli nazwa jest mniejsza lub równa 28 bajtów, to nazwa jest wstawiana, dopełniona do prawej strony spacjami.

Jeśli nazwa jest większa niż 28 bajtów, wstawiana jest najmniejsza liczba 28 bajtów nazwy.

Linux i Solaris

Jeśli nazwa jest mniejsza lub równa 15 bajtów, to nazwa jest wstawiana, dopełniona do prawej strony spacjami.

Jeśli nazwa jest większa niż 15 bajtów, wstawiane są maksymalnie 15 bajtów nazwy, dopełniane do prawej strony spacjami.

HP-UX

Jeśli nazwa jest mniejsza lub równa 14 bajtów, to nazwa jest wstawiana, dopełniona do prawej strony spacjami.

Jeśli nazwa jest większa niż 14 bajtów, wstawiane są maksymalnie 14 bajtów nazwy, dopełniane do prawej strony spacjami.

Na przykład w przypadku uruchomienia produktu /opt/mqm/samp/bin/amqspuT QNAME QMNAME nazwa PutAppl będzie mieć wartość 'amqspuT'. W tym polu MQCHAR28 znajduje się 21 znaków spacji. Należy zauważyć, że pełna ścieżka, w tym /opt/mqm/samp/bin, nie jest zawarta w nazwie PutAppl.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PutMsgOpts* podano wartość MQPMO_SET_ALL_CONTEXT. Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli parametr MQPMO_SET_ALL_CONTEXT nie został określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Typ PutAppl(MQLONG)

Jest to typ aplikacji umieszczonej w komunikacie i jest częścią **kontekstu pochodzenia** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i Kontekst komunikatu.

Produkt *PutApplType* może mieć jeden z następujących typów standardowych. Możliwe jest również zdefiniowanie własnych typów, ale tylko z wartościami z zakresu MQAT_USER_FIRST za pomocą MQAT_USER_LAST.

MQAT_AIX

Aplikacja AIX (ta sama wartość jak MQAT_UNIX).

V 8.0.0.4 MQAT_AMQP

Aplikacja protokołu AMQP

MQAT_BROKER

Broker.

MQAT_CICS

CICS .

MQAT_CICS_BRIDGE

CICS bridge.

MQAT_CICS_VSE

CICS/VSE .

MQAT_DOS

Aplikacja IBM MQ MQI client na komputerze PC DOS.

MQAT_DQM

Agent menedżera kolejek rozproszonych.

MQAT_GUARDIAN,

Tandem Guardian aplikacja (ta sama wartość jak MQAT_NSK).

MQAT_IMS

Aplikacja IMS .

MQAT_IMS_BRIDGE

Most IMS .

MQAT_JAVA

Java.

MQAT_MVS

MVS lub aplikacji TSO (taka sama wartość jak MQAT_ZOS).

MQAT_NOTES_AGENT

Lotus Notes Aplikacja agenta.

MQAT_NSK

Aplikacja HP Integrity NonStop Server .

MQAT_OS390

Aplikacja OS/390 (taka sama wartość jak MQAT_ZOS).

MQAT_OS400

Aplikacja IBM i .

MQAT_QMGR

menedżerze kolejek.

MQAT_UNIX

Aplikacja UNIX .

MQAT_VOS

Aplikacja Stratus VOS.

MQAT_WINDOWS

16-bitowa aplikacja Windows .

MQAT_WINDOWS_NT

32-bitowa aplikacja Windows .

MQAT_WLM

Aplikacja menedżera obciążenia produktu z/OS .

MQAT_XCF

XCF.

MQAT_ZOS

Aplikacja z/OS .

MQAT_DEFAULT

Domyślny typ aplikacji.

Jest to domyślny typ aplikacji dla platformy, na której działa aplikacja.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska. Z tego powodu zawsze należy skompilować aplikację, używając w tym celu plików nagłówkowych, include lub COPY odpowiednich dla platformy, na której będzie uruchamiana aplikacja.

MQAT_UNKNOWN

Tej wartości należy użyć, aby wskazać, że typ aplikacji jest nieznany, mimo że występują inne informacje o kontekście.

MQAT_USER_FIRST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

MQAT_USER_LAST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Może wystąpić również następująca wartość specjalna:

MQAT_NO_CONTEXT

Ta wartość jest ustawiana przez menedżer kolejek, gdy komunikat jest umieszczany bez kontekstu (to znaczy, że określono opcję kontekstu MQPMO_NO_CONTEXT).

Po pobraniu komunikatu program *PutApplType* może zostać przetestowany pod kątem tej wartości, aby określić, czy komunikat ma kontekst (zaleca się, aby produkt *PutApplType* nigdy nie był ustawiony na wartość MQAT_NO_CONTEXT przez aplikację używającą MQPMO_SET_ALL_CONTEXT, jeśli którykolwiek z pozostałych pól kontekstu jest niepusty).

Gdy menedżer kolejek generuje te informacje w wyniku umieszczenia aplikacji, pole jest ustawiane na wartość, która jest określana przez środowisko. W systemie IBM i jest ona ustawiona na wartość MQAT_OS400; menedżer kolejek nigdy nie używa komendy MQAT_CICS w systemie IBM i.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PutMsgOpts* podano wartość MQPMO_SET_ALL_CONTEXT. Jeśli parametr MQPMO_SET_ALL_CONTEXT nie został określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

To jest pole wyjściowe dla wywołania MQGET. Początkowa wartość tego pola to MQAT_NO_CONTEXT.

PutDate (MQCHAR8)

Jest to data umieszczenia komunikatu i jest częścią **kontekstu pochodzenia** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD” na stronie 418](#) i [Kontekst komunikatu](#).

Format używany dla daty, w której to pole jest generowane przez menedżer kolejek:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr

rok (cztery cyfry)

MM

miesiąc w roku (01 do 12)

DD

Dzień miesiąca (01 do 31)

Czas Greenwich (GMT) jest używany dla pól *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, data jest datą umieszczenia komunikatu, a nie datą, kiedy jednostka pracy została zatwierdzona.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PutMsgOpts* podano wartość MQPMO_SET_ALL_CONTEXT. Zawartość pola nie jest sprawdzana przez menedżer kolejek, z tym wyjątkiem, że wszystkie informacje po znaku NULL w tym polu są usuwane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące znaki w puste znaki. Jeśli parametr MQPMO_SET_ALL_CONTEXT nie został określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez wartość MQ_PUT_DATE_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C, a 8 znaków odstępu w innych językach programowania.

PutTime (MQCHAR8)

Jest to czas umieszczenia komunikatu, który jest częścią **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i [Kontekst komunikatu](#).

Format używany w czasie, gdy pole jest generowane przez menedżer kolejek:

- GGMMSSSTH

gdzie znaki reprezentują (w kolejności):

GG

godzin (od 00 do 23)

MM

minuty (od 00 do 59)

SS

sekundy (od 00 do 59; patrz uwaga)

T

Dziesiątych sekundy (od 0 do 9)

H

Setnych części sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, można w rzadkich przypadkach zwrócić 60 lub 61 sekund w ciągu sekundy w składce *PutTime*. Dzieje się tak wtedy, gdy sekundy przestępne są wstawiane w globalnym standardzie czasu.

Czas Greenwich (GMT) jest używany dla pól *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, godzina jest taka, gdy komunikat został wstawiony, a nie czas, w którym jednostka pracy została zatwierdzona.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PutMsgOpts* podano wartość MQPMO_SET_ALL_CONTEXT. Menedżer kolejek nie sprawdza zawartości tego pola, z tym wyjątkiem, że wszystkie informacje po znaku null w tym polu są usuwane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące znaki w puste znaki. Jeśli parametr MQPMO_SET_ALL_CONTEXT nie został określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez wartość MQ_PUT_TIME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C, a 8 znaków odstępu w innych językach programowania.

ReplyToQ (MQCHAR48)

Jest to nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania dla komunikatu, wysyła komunikaty MQMT_REPLY i MQMT_REPORT. Nazwa to lokalna nazwa kolejki zdefiniowana w menedżerze kolejek identyfikowana przez produkt *ReplyToQMgr*. Ta kolejka nie może być kolejką modelową, chociaż menedżer kolejek wysyłających nie weryfikuje tej kolejki po umieszczeniu w niej komunikacie.

W przypadku wywołań MQPUT i MQPUT1 pole to nie może być puste, jeśli pole *MsgType* ma wartość MQMT_REQUEST, lub jeśli żądane są komunikaty raportu w polu *Report* (Żądanie). Jednak podana wartość (lub podstawiona) jest przekazywana do aplikacji, która wydaje żądanie pobrania dla komunikatu, niezależnie od typu komunikatu.

Jeśli pole *ReplyToQMgr* jest puste, lokalny menedżer kolejek będzie wyszukiwać nazwę *ReplyToQ* we własnych definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *ReplyToQ* w przekazanej wiadomości jest zastępowana wartością atrybutu *RemoteQName* z definicji kolejki zdalnej, a ta wartość jest zwracana w deskrytorze komunikatu, gdy aplikacja odbierający wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, *ReplyToQ* pozostaje niezmienną.

Jeśli nazwa jest określona, może zawierać odstępy końcowe; pierwszy pusty znak i znaki po nim są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane sprawdzanie, czy nazwa spełnia reguły nazewnictwa dla kolejek. Jest to również prawdziwe w przypadku nadawanej nazwy, jeśli *ReplyToQ* jest zastępowana w przesyłanej wiadomości. Jedynym sprawdzany jest fakt, że podano nazwę, jeśli wymagają tego okoliczności.

Jeśli kolejka zwrotna nie jest wymagana, ustaw pole *ReplyToQ* na puste lub (w języku programowania C) na łańcuch pusty lub na jeden lub więcej znaków odstępu, po których następuje znak o kodzie zero; nie pozostaw pola niezainicjowanego.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnioną spacjami do długości pola.

Jeśli komunikat, który wymaga komunikatu raportu, nie może zostać dostarczony, a komunikat raportu również nie może zostać dostarczony do określonej kolejki, to zarówno oryginalny komunikat, jak i komunikat raportu są wyświetlane w kolejce niedostarczonych komunikatów (patrz atrybut *DeadLetterQName* opisany w sekcji „Atrybuty dla menedżera kolejek” na stronie 810).

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

ReplyToQMgr (MQCHAR48)

Jest to nazwa menedżera kolejek, do którego ma zostać wysłany komunikat odpowiedzi lub komunikat raportu. *ReplyToQ* to nazwa lokalna kolejki, która jest zdefiniowana w tym menedżerze kolejek.

Jeśli pole *ReplyToQMgr* jest puste, lokalny menedżer kolejek wyszuka nazwę *ReplyToQ* w definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *ReplyToQMgr* w przekazanej wiadomości jest zastępowana wartością atrybutu *RemoteQMgrName* z definicji kolejki zdalnej, a ta wartość jest zwracana w deskrytorze komunikatu, gdy aplikacja odbierający wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, *ReplyToQMgr*, która jest przesyłana z komunikatem, jest nazwą lokalnego menedżera kolejek.

Jeśli nazwa jest określona, może zawierać odstępy końcowe; pierwszy pusty znak i znaki po nim są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane sprawdzanie, czy nazwa spełnia reguły nazewnictwa dla menedżerów kolejek lub czy ta nazwa jest znana wysyłającemu menedżerowi kolejek. Jest to również prawda dla przesyłanych nazw, jeśli *ReplyToQMgr* jest zastępowana w przekazanej wiadomości.

Jeśli kolejka zwrotna nie jest wymagana, ustaw pole *ReplyToQMgr* na puste lub (w języku programowania C) na łańcuch pusty lub na jeden lub więcej znaków odstępu, po których następuje znak o kodzie zero; nie pozostaw pola niezainicjowanego.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnioną spacjami do długości pola.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Raport (MQLONG)

Komunikat raportu to komunikat o innym komunikacie, który jest używany do informowania aplikacji o oczekiwanych lub nieoczekiwanych zdarzeniach, które odnoszą się do oryginalnego komunikatu. Pole *Report* umożliwia aplikacji wysyłanie oryginalnego komunikatu w celu określenia, które komunikaty raportów są wymagane, czy dane komunikatu aplikacji mają być zawarte w nich, a także (dla obu raportów i odpowiedzi), w jaki sposób mają być ustawione identyfikatory komunikatów i korelacji w komunikacie raportu lub odpowiedzi. Można zażądać dowolnego lub wszystkiego (lub żadnego) z następujących typów komunikatów raportu:

- Wyjątek
- Termin ważności
- Potwierdź po przybyciu (COA)
- Potwierdzenie dostarczenia (COD)
- Powiadomienie o działaniu pozytywnym (PAN)
- Powiadomienie o działaniu negatywnym (NAN)

Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Aplikacja, która odbiera komunikat raportu, może określić przyczynę wygenerowania raportu, sprawdzając pole *Feedback* w strukturze MQMD. Więcej szczegółów zawiera pole *Feedback*.

Użycie opcji raportu podczas umieszczania komunikatu w temacie może spowodować wygenerowanie i wysłanie do aplikacji wartości zero, jeden lub wiele komunikatów raportu. Jest to spowodowane tym, że komunikat publikacji może być wysyłany do aplikacji o wartości zero, jednej lub wielu aplikacjach subskrybujących.

Opcje wyjątku: należy określić jedną z opcji wymienionych w celu żądania komunikatu o wyjątku.

MQRO_EXCEPTION

Agent kanału komunikatów generuje ten typ raportu, gdy komunikat jest wysyłany do innego menedżera kolejek, a komunikat nie może zostać dostarczony do określonej kolejki docelowej. Na przykład kolejka docelowa lub pośrednia kolejka transmisji może być pełna, albo komunikat może być zbyt duży dla kolejki.

Generowanie komunikatu raportu o wyjątkach zależy od trwałości oryginalnego komunikatu oraz szybkości kanału komunikatów (normalnego lub szybkiego), za pośrednictwem którego oryginalny komunikat jest przemieszczany:

- W przypadku wszystkich komunikatów trwałych oraz w przypadku komunikatów nietrwałych podróżujących za pośrednictwem zwykłych kanałów komunikatów raport o wyjątku jest generowany *tylko*, jeśli działanie określone przez aplikację wysyłającą dla warunku błędu może zostać pomyślnie zakończone. Aplikacja wysyłająca może określić jedno z następujących działań, aby sterować rozporządzaniem oryginalnego komunikatu w przypadku wystąpienia warunku błędu:
 - MQRO_DEAD_LETTER_Q (to umieszcza oryginalną wiadomość w kolejce niedostarczonych komunikatów).
 - MQRO_DISCARD_MSG (powoduje to usunięcie oryginalnego komunikatu).

Jeśli działanie określone przez aplikację wysyłającą nie może zostać zakończone pomyślnie, oryginalny komunikat jest pozostawiony w kolejce transmisji i nie zostanie wygenerowany żaden komunikat o raporcie o wyjątku.

- W przypadku komunikatów nietrwałych podróżujących za pomocą szybkich kanałów komunikatów oryginalny komunikat jest usuwany z kolejki transmisji, a raport wyjątku wygenerowany *nawet wtedy, gdy* nie można pomyślnie zakończyć określonego działania dla warunku błędu. Na przykład, jeśli określono wartość MQRO_DEAD_LETTER_Q, ale oryginalny komunikat nie może zostać umieszczony w kolejce niedostarczonych komunikatów, ponieważ ta kolejka jest pełna, generowany jest komunikat o wyjątku i oryginalny komunikat został usunięty.

Więcej informacji na temat normalnych i szybkich kanałów komunikatów zawiera sekcja Szybkość komunikatów nietrwałych (NPMSPEED).

Raport o wyjątku nie jest generowany, jeśli aplikacja, która umieściła oryginalny komunikat, może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1.

Aplikacje mogą również wysyłać raporty o wyjątkach, aby wskazać, że komunikat nie może być przetworzony (na przykład, ponieważ jest to transakcja debetowa, która powodowałaby przekroczenie limitu kredytowego rachunku).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy określać więcej niż jednego z następujących wartości: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA

Jest to takie samo, jak w przypadku wyjątku MQRO_EXCEPTION, z tym wyjątkiem, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie należy określać więcej niż jednego z następujących wartości: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA

Raporty o wyjątkach z pełnymi danymi są wymagane.

Jest to takie samo, jak w przypadku wyjątku MQRO_EXCEPTION, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie należy określać więcej niż jednego z następujących wartości: MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

Opcje utraty ważności: należy podać jedną z opcji wymienionych w celu zażądania komunikatu o utracie ważności raportu.

MQRO_EXPIRATION

Ten typ raportu jest generowany przez menedżer kolejek, jeśli komunikat został odrzucony przed dostarczeniu do aplikacji, ponieważ upłynął jego czas utraty ważności (patrz pole *Expiry*). Jeśli ta opcja nie zostanie ustawiona, żaden komunikat raportu nie zostanie wygenerowany, jeśli komunikat zostanie usunięty z tego powodu (nawet jeśli zostanie podany jeden z opcji MQRO_EXCEPTION_*).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA

Jest to taka sama sytuacja jak MQRO_EXPIRATION, z tym wyjątkiem, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA

Jest to taka sama sytuacja jak MQRO_EXPIRATION, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

Opcje potwierdzania przybycia: Określ jedną z opcji, które mają zostać wyświetlone w celu zażądania komunikatu potwierdzenia w momencie przybycia.

MQRO_COA

Ten typ raportu jest generowany przez menedżer kolejek, który jest właścicielem kolejki docelowej, gdy komunikat jest umieszczany w kolejce docelowej. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest umieszczany jako część jednostki pracy, a kolejka docelowa jest kolejką lokalną, komunikat raportu COA wygenerowany przez menedżer kolejek może zostać pobrany tylko wtedy, gdy jednostka pracy jest zatwierdzona.

Raport COA nie jest generowany, jeśli pole *Format* w deskrypcorze komunikatu ma wartość MQFMT_XMIT_Q_HEADER lub MQFMT_DEAD_LETTER_HEADER. Zapobiega to generowaniu raportu COA, jeśli komunikat jest umieszczany w kolejce transmisji, lub jest niedostarczalny i umieszczony w kolejce niedostarczonych komunikatów.

W przypadku kolejki mostu IMS raport COA jest generowany, gdy komunikat dociera do kolejki produktu IMS (potwierdzenie odebrane z IMS) i nie, gdy komunikat jest umieszczany w kolejce mostu MQ. Oznacza to, że jeśli produkt IMS nie jest aktywny, raport COA nie jest generowany do momentu uruchomienia produktu IMS, a komunikat jest umieszczany w kolejce w kolejce IMS.

Użytkownik, który uruchamia program, który umieszcza komunikat MQMD.Report= MQRO_COA musi mieć uprawnienie + passid w kolejce odpowiedzi. Jeśli użytkownik nie ma uprawnień + passid, komunikat raportu COA nie dociska do kolejki odpowiedzi. Podjęto próbę umieszczenia komunikatu raportu w kolejce niedostarczanych komunikatów.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_DATA

Jest to takie samo, jak MQRO_COA, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_FULL_DATA

Jest to takie samo, jak MQRO_COA, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie należy określać więcej niż jednej z następujących wartości: MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

Opcje potwierdzania po dostarczeniu: Określ jedną z opcji, które mają zostać wyświetlone w celu zażądania komunikatu potwierdzania dotyczącego dostarczenia.

MQRO_COD

Ten typ raportu jest generowany przez menedżer kolejek, gdy aplikacja pobiera komunikat z kolejki docelowej w sposób, który usuwa komunikat z kolejki. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest pobierany jako część jednostki pracy, komunikat raportu jest generowany w ramach tej samej jednostki pracy, tak aby raport nie był dostępny do momentu zatwierdzenia jednostki pracy. Jeśli jednostka pracy jest wycofana, raport nie jest wysyłany.

Raport COD nie zawsze jest generowany, jeśli komunikat jest pobierany za pomocą opcji MQGMO_MARK_SKIP_BACKOUT. Jeśli tworzona jest kopia zapasowa podstawowej jednostki pracy, ale dodatkowa jednostka pracy jest zatwierdzana, komunikat jest usuwany z kolejki, ale raport COD nie jest generowany.

Raport COD nie jest generowany, jeśli pole *Format* w deskrypcorze komunikatu ma wartość MQFMT_DEAD_LETTER_HEADER. Zapobiega to generowaniu raportu COD, jeśli komunikat jest niedostarczalny i jest umieszczany w kolejce niedostarczonych komunikatów.

Parametr MQRO_COD nie jest poprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jednej z następujących wartości: MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_DATA

Jest to ta sama wartość co MQRO_COD, z tym wyjątkiem, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Jeśli w wywołaniu MQGET dla oryginalnego komunikatu określono wartość MQGMO_ACCEPT_TRUNCATED_MSG, a wczytany komunikat jest obcięty, ilość danych komunikatu aplikacji umieszczanych w komunikacie raportu zależy od środowiska:

- W systemie z/OS jest to minimum:
 - Długość oryginalnego komunikatu
 - Długość buforu użytego do pobrania komunikatu.
 - 100 bajtów.
- W innych środowiskach jest to minimum:
 - Długość oryginalnego komunikatu
 - 100 bajtów.

Parametr MQRO_COD_WITH_DATA nie jest poprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jednej z następujących wartości: MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_FULL_DATA

Jest to taka sama sytuacja, jak w przypadku wywołania MQRO_COD, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

MQRO_COD_WITH_FULL_DATA nie jest poprawne, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jednej z następujących wartości: MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

Opcje powiadamiania o działaniu: należy określić jedną lub obie opcje wymienione w celu żądania wysłania przez aplikację odbierającą komunikatu o pozytywnym działaniu lub komunikatu o negatywnym działaniu.

MQRO_PAN

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie zostało wykonane pomyślnie. Aplikacja generujący raport określa, czy dane mają zostać dołączone do raportu.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby aplikacja pobierający musi wygenerować raport.

MQRO_NAN

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie *nie* zostało wykonane pomyślnie. Aplikacja generujący raport

określa, czy dane mają zostać dołączone do raportu. Można na przykład uwzględnić niektóre dane wskazujące, dlaczego żądanie nie mogło zostać wykonane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby aplikacja pobierająca musi wygenerować raport.

Wniosek musi ustalić, które warunki odpowiadają pozytywnym działaniom i które odpowiadają negatywnym działaniom. Jeśli jednak żądanie zostało wykonane tylko częściowo, wygeneruj raport NAN, a nie raport PAN, jeśli jest to wymagane. Każdy możliwy warunek musi być zgodny z działaniem pozytywnym lub działaniem negatywnym, ale nie musi być spełniony jednocześnie.

Opcje identyfikatora komunikatu: należy podać jedną z opcji wymienionych w celu kontrolowania sposobu, w jaki *MsgId* ma być ustawiony komunikat raportu (lub komunikat odpowiedzi).

MQRO_NEW_MSG_ID

Jest to działanie domyślne, które wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, dla raportu lub odpowiedzi zostanie wygenerowany nowy *MsgId*.

MQRO_PASS_MSG_ID

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, *MsgId* tego komunikatu jest kopiowany do *MsgId* komunikatu raportu lub odpowiedzi.

MsgId komunikatu publikacji będzie się różnić dla każdego subskrybenta, który otrzymuje kopię publikacji, dlatego *MsgId* skopiowana do raportu lub komunikat odpowiedzi będzie się różnić dla każdego z tych publikacji.

Jeśli ta opcja nie zostanie podana, przyjmowana jest wartość *MQRO_NEW_MSG_ID*.

Opcje identyfikatora korelacji: należy określić jedną z opcji wymienionych w celu kontrolowania sposobu, w jaki *CorrelId* ma być ustawiony komunikat raportu (lub komunikat odpowiedzi).

MQRO_COPY_MSG_ID_TO_CORREL_ID (Identyfikator CORREL_ID)

Jest to działanie domyślne, które wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, *MsgId* tego komunikatu jest kopiowany do *CorrelId* komunikatu raportu lub odpowiedzi.

MsgId komunikatu publikacji będzie się różnić dla każdego subskrybenta, który otrzymuje kopię publikacji, dlatego *MsgId* skopiowana do *CorrelId* komunikatu raportu lub odpowiedzi będzie się różnić dla każdego z nich.

MQRO_PASS_CORREL_ID

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, *CorrelId* tego komunikatu jest kopiowany do *CorrelId* komunikatu raportu lub odpowiedzi.

CorrelId komunikatu publikacji będzie specyficzny dla subskrybenta, chyba że użyje opcji *MQSO_SET_CORREL_ID* i ustawi pole identyfikatora *SubCorrelw* tabeli *MQSD* na *MQCI_NONE*. Z tego powodu możliwe jest, że *CorrelId* skopiowana do *CorrelId* komunikatu raportu lub odpowiedzi będzie inna dla każdego z nich.

Jeśli ta opcja nie zostanie podana, przyjmowana jest wartość *MQRO_COPY_MSG_ID_TO_CORREL_ID*.

Serwery odpowiadające na żądania lub generujące komunikaty raportów muszą sprawdzić, czy w pierwotnym komunikacie zostały ustawione opcje *MQRO_PASS_MSG_ID* lub *MQRO_PASS_CORREL_ID*. Jeśli były, serwery muszą wykonać działanie opisane dla tych opcji. Jeśli żadna z tych wartości nie zostanie ustawiona, serwery muszą wykonać odpowiednie działanie domyślne.

Opcje rozporządzania: Określ jedną z wyświetlonych opcji, aby sterować rozporządzaniem oryginalnego komunikatu, gdy nie można go dostarczyć do kolejki docelowej. Aplikacja może ustawić opcje rozporządzania niezależnie od żądania raportów o wyjątkach.

MQRO_DEAD_LETTER_Q

Jest to działanie domyślne, które umieszcza komunikat w kolejce niedostarczonych komunikatów, jeśli komunikat nie może zostać dostarczony do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Generowany jest komunikat o wyjątku, o ile został on zażądany przez nadawcę.
- Gdy aplikacja, która umieściła oryginalny komunikat, została wstawiona do tematu

MQRO_DISCARD_MSG

Spowoduje to usunięcie komunikatu, jeśli nie może zostać dostarczony do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Generowany jest komunikat o wyjątku, o ile został on zażądany przez nadawcę.
- Gdy aplikacja, która umieściła oryginalny komunikat, została wstawiona do tematu

Jeśli oryginalny komunikat ma zostać zwrócony do nadawcy, a oryginalny komunikat nie jest umieszczany w kolejce niedostarczonych komunikatów, nadawca musi określić MQRO_DISCARD_MSG z MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_AND_WAŻNOŚCI

Jeśli ta opcja jest ustawiona w komunikacie, a raport lub odpowiedź jest generowana z powodu tego komunikatu, deskryptor komunikatu dziedziczy:

- MQRO_DISCARD_MSG, jeśli został ustawiony.
- Pozostały czas utraty ważności komunikatu (jeśli nie jest to raport utraty ważności). Jeśli jest to raport utraty ważności, czas utraty ważności jest ustawiony na 60 sekund.

Opcja działania

MQRO_ACTIVITY,

Użycie tej wartości pozwala na śledzenie trasy **dowolnego** komunikatu w całej sieci menedżera kolejek. Opcja raportu może być określona w dowolnym komunikacie bieżącego użytkownika, co pozwala na natychmiastowe rozpoczęcie obliczania trasy wiadomości przez sieć.

Jeśli aplikacja generowana przez komunikat nie może przełączać się na raporty aktywności, raporty mogą być włączone przy użyciu wyjścia funkcji API, które jest dostarczane przez administratorów menedżera kolejek.

Uwaga:

1. Im mniej menedżerów kolejek w sieci, które są w stanie generować raporty aktywności, tym mniej szczegółowa trasa jest szczegółowa.
2. Raporty dotyczące działań mogą być trudne do umieszczenia w poprawnej kolejności w celu określenia podjętej trasy.
3. Raporty dotyczące działań mogą nie być w stanie znaleźć trasy dożądanego miejsca docelowego.
4. Komunikaty z tym zestawem opcji raportu muszą być akceptowane przez dowolnego menedżera kolejek, nawet jeśli nie rozumieją tej opcji. Pozwala to na ustawienie opcji raportu dla dowolnego komunikatu użytkownika, nawet jeśli są one przetwarzane przez menedżer kolejek inny niż Wersja 6.0 lub nowszy.
5. Jeśli proces, menedżer kolejek lub proces użytkownika, wykonuje działanie na komunikacie z tym zestawem opcji, może on zdecydować o wygenerowaniu i umieszczeniu raportu aktywności.

Opcja domyślna: należy podać następujące opcje, jeśli nie są wymagane żadne opcje raportu:

MQRO_NONE

Użyj tej wartości, aby wskazać, że nie określono żadnych innych opcji. Parametr MQRO_NONE jest zdefiniowany w celu uzyskania dokumentacji programu pomocy. Ta opcja nie jest przeznaczona do użycia z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Informacje ogólne:

1. Wszystkie wymagane typy raportów muszą być specjalnie wymagane przez aplikację wysyłając oryginalny komunikat. Na przykład, jeśli zażądano raportu COA, ale raport o wyjątku nie jest, raport COA jest generowany, gdy komunikat jest umieszczany w kolejce docelowej, ale nie jest generowany

żaden raport o wyjątku, jeśli kolejka docelowa jest zapełniona po przybyciu komunikatu do kolejki docelowej. Jeśli nie ustawiono opcji *Report*, menedżer kolejek lub agent kanału komunikatów (MCA) nie wygeneruje komunikatów raportu.

Niektóre opcje raportu można określić nawet wtedy, gdy lokalny menedżer kolejek nie rozpoznaje ich. Jest to przydatne, gdy opcja ma być przetwarzana przez menedżer kolejek *docelowy*. Więcej szczegółów na ten temat zawiera sekcja „Opcje raportów i flagi komunikatów” na stronie 915.

Jeśli zażądano komunikatu raportu, nazwa kolejki, do której ma zostać wysłany raport, musi być określona w polu *ReplyToQ*. Po odebraniu komunikatu z raportem rodzaj raportu można określić, badając pole *Feedback* w deskrypcji komunikatu.

2. Jeśli menedżer kolejek lub agent MCA generujący komunikat raportu nie może umieścić komunikatu raportu w kolejce odpowiedzi (na przykład dlatego, że kolejka odpowiedzi lub kolejka transmisji jest pełna), komunikat raportu zostanie umieszczony w kolejce niedostarczonych komunikatów. Jeśli również nie powiedzie się, lub nie ma kolejki niedostarczonych komunikatów, działanie jest zależne od typu komunikatu raportu:
 - Jeśli komunikat raportu jest raportem o wyjątkach, komunikat, który wygenerował raport o wyjątkach, pozostaje w swojej kolejce transmisji; zapewnia to, że komunikat nie zostanie utracony.
 - Dla wszystkich pozostałych typów raportów komunikat raportu jest odrzucany, a przetwarzanie jest kontynuowane normalnie. Dzieje się tak dlatego, że oryginalny komunikat został już bezpiecznie dostarczony (dla komunikatów raportu COA lub COD) lub nie jest już zainteresowany (dla komunikatu o utracie ważności).

Gdy komunikat raportu zostanie pomyślnie umieszczony w kolejce (kolejka docelowa lub pośrednia kolejka transmisji), komunikat nie będzie już podlegał specjalnym przetwarzaniu; jest traktowany tak samo jak każdy inny komunikat.

3. Po wygenerowaniu raportu kolejka *ReplyToQ* jest otwierana, a komunikat raportu jest umieszczany przy użyciu uprawnień *UserIdentifier* w strukturze MQMD komunikatu, co powoduje, że raport jest generowany, z wyjątkiem następujących przypadków:
 - Raporty o wyjątkach wygenerowane przez odbierający agent MCA są umieszczane z dowolnymi uprawnieniami używanego przez agenta MCA podczas próby umieszczenia komunikatu powodującego raport.
 - Raporty COA wygenerowane przez menedżera kolejek są umieszczane z dowolnymi uprawnieniami, gdy komunikat powodujący wygenerowanie raportu został wygenerowany przez menedżer kolejek generujący raport. Na przykład, jeśli komunikat został umieszczony przez odbierającego agenta MCA przy użyciu identyfikatora użytkownika MCA, menedżer kolejek umieszcza raport COA przy użyciu identyfikatora użytkownika MCA.

Aplikacje generujące raporty muszą korzystać z tego samego uprawnienia, co w przypadku generowania odpowiedzi. Zwykle jest to uprawnienie identyfikatora użytkownika w oryginalnym komunikacie.

Jeśli raport musi podróżować do miejsca docelowego, nadawcy i odbiorcy mogą zdecydować, czy go zaakceptować, w taki sam sposób, jak w przypadku innych komunikatów.

4. Jeśli zażądano komunikatu raportu z danymi:
 - Komunikat raportu jest zawsze generowany wraz z ilością danych żądanych przez nadawcę oryginalnego komunikatu. Jeśli komunikat raportu jest zbyt duży dla kolejki odpowiedzi, przetwarzanie opisane powyżej jest wykonywane. Komunikat raportu nigdy nie jest obcinany, aby zmieścić się w kolejce odpowiedzi.
 - Jeśli *Format* oryginalnego komunikatu to MQFMT_XMIT_Q_HEADER, dane zawarte w raporcie nie zawierają tabeli MQXQH. Dane raportu rozpoczynają się od pierwszego bajtu danych poza MQXQH w oryginalnym komunikacie. Dzieje się tak, niezależnie od tego, czy kolejka jest kolejką transmisji.
5. Jeśli w kolejce odpowiedzi zostanie odebrany komunikat COA, COD lub raportu o utracie ważności, to zostanie zagwarantowane, że oryginalny komunikat został dostarczony, został dostarczony lub utracił ważność, w zależności od przypadku. Jeśli jednak zażądano co najmniej jednego z tych komunikatów

raportu, a *nie* zostanie odebrany, nie można założyć odwrotnej sytuacji, ponieważ mogło wystąpić jedno z następujących zdarzeń:

- a. Komunikat raportu jest wstrzymany, ponieważ odsyłacz jest wyłączony.
- b. Komunikat raportu jest wstrzymany, ponieważ warunek blokowania istnieje w pośredniej kolejce transmisji lub w kolejce odpowiedzi (na przykład kolejka jest zapelniona lub zablokowana dla operacji put).
- c. Komunikat raportu znajduje się w kolejce niedostarczonych komunikatów.
- d. Gdy menedżer kolejek próbował wygenerować komunikat raportu, nie mógł on umieścić go w odpowiedniej kolejce ani w kolejce niedostarczonych komunikatów, dlatego nie można było wygenerować komunikatu raportu.
- e. Wystąpił błąd menedżera kolejek między raportowaniem działania (nadejściem, dostawą lub utratą ważności) a wygenerowaniem odpowiedniego komunikatu raportu. (Nie zdarza się to dla komunikatów raportu COD, jeśli aplikacja wczytuje oryginalny komunikat w jednostce pracy, ponieważ komunikat raportu COD jest generowany w ramach tej samej jednostki pracy).

Komunikaty raportów o wyjątkach mogą być przechowywane w taki sam sposób, jak przyczyny 1, 2 i 3 powyżej. Jeśli jednak agent MCA nie może wygenerować komunikatu raportu o wyjątkach (komunikat raportu nie może zostać umieszczony w kolejce odpowiedzi lub w kolejce niedostarczonych komunikatów), oryginalny komunikat pozostaje w kolejce transmisji u nadawcy, a kanał jest zamknięty. Dzieje się tak niezależnie od tego, czy komunikat raportu miał być generowany podczas wysyłania, czy też odbierającego końca kanału.

6. Jeśli oryginalny komunikat jest tymczasowo zablokowany (w wyniku czego generowany jest komunikat o wyjątku, a oryginalny komunikat jest umieszczany w kolejce niedostarczonych komunikatów), ale blokada i aplikacja odczyta oryginalny komunikat z kolejki niedostarczonych komunikatów i umieszcza go ponownie w miejscu docelowym, mogą wystąpić następujące działania:
 - Mimo że wygenerowano komunikat o wyjątku, oryginalny komunikat w końcu dotarł do miejsca docelowego.
 - W odniesieniu do pojedynczego oryginalnego komunikatu generowany jest więcej niż jeden komunikat raportu o wyjątku, ponieważ pierwotny komunikat może napotkać inną blokadę w późniejszym czasie.

Zgłaszanie komunikatów podczas wprowadzania do tematu:

1. Raporty mogą być generowane podczas umieszczania komunikatu w temacie. Ten komunikat zostanie wysłany do wszystkich subskrybentów tematu, który może być równy zero, jeden lub wiele. Należy to wziąć pod uwagę przy wyborze opcji raportu, ponieważ w rezultacie można wygenerować wiele komunikatów raportu.
2. Podczas umieszczania komunikatu w temacie może istnieć wiele kolejek docelowych, które mają zostać nadane kopii komunikatu. Jeśli niektóre z tych kolejek docelowych mają problem, na przykład zapewnianie kolejki, pomyślne zakończenie operacji MQPUT jest uzależnione od ustawienia wartości NPMSGDLV lub PMSGDLV (w zależności od trwałości komunikatu). Jeśli to ustawienie jest takie, że dostarczenie komunikatu do kolejki docelowej musi być pomyślne (na przykład jest to komunikat trwały dla trwałego subskrybenta, a parametr PMSGDLV jest ustawiony na ALL lub ALLDUR), powodzenie jest definiowane jako jedno z następujących kryteriów:
 - Pomyślnie umieszczono w kolejce subskrybenta
 - Użycie komendy MQRO_DEAD_LETTER_Q i pomyślnej operacji umieszczania w kolejce niedostarczonych komunikatów, jeśli kolejka subskrybenta nie może odebrać komunikatu.
 - Użycie komendy MQRO_DISCARD_MSG, jeśli kolejka subskrybenta nie może odebrać komunikatu.

Komunikaty raportów dla segmentów komunikatów:

1. Komunikaty raportów mogą być wymagane dla komunikatów, które mają dozwolone segmentację (patrz opis opcji MQMF_SEGMENTATION_ALLOWED). Jeśli menedżer kolejek stwierdzi, że konieczne jest segmentowanie komunikatu, może zostać wygenerowany komunikat raportu dla każdego z segmentów, które następnie napotka odpowiedni warunek. Aplikacje muszą być przygotowane do odbierania wielu komunikatów raportu dla każdego żadanego typu komunikatu raportu. Użyj pola

GroupId w komunikacie raportu, aby skorelować wiele raportów z identyfikatorem grupy oryginalnego komunikatu, a pole *Feedback* zidentyfikuj typ każdego komunikatu raportu.

2. Jeśli komenda MQGMO_LOGICAL_ORDER jest używana do pobierania komunikatów raportu dla segmentów, należy pamiętać, że raporty o *różnych typach* mogą być zwracane przez kolejne wywołania MQGET. Na przykład, jeśli żądane są zarówno raporty COA, jak i COD dla komunikatu posegmentowanego przez menedżer kolejek, wywołania MQGET dla komunikatów raportu mogą zwrócić komunikaty raportu COA i COD interleaved w nieprzewidywalny sposób. Należy unikać tego, używając opcji MQGMO_COMPLETE_MSG (opcjonalnie z opcją MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG powoduje, że menedżer kolejek ponownie składa komunikaty raportu, które mają ten sam typ raportu. Na przykład pierwsze wywołanie MQGET może ponownie złożyć wszystkie komunikaty COA odnoszące się do oryginalnego komunikatu, a drugie wywołanie MQGET może ponownie złożyć wszystkie komunikaty COD. Najpierw zmontowany najpierw zależy od tego, który typ komunikatu raportu pojawia się jako pierwszy w kolejce.
3. Aplikacje, które samodzielnie umieszczają segmenty, mogą określać różne opcje raportów dla każdego segmentu. Należy jednak zwrócić uwagę na następujące kwestie:
 - Jeśli segmenty są pobierane przy użyciu opcji MQGMO_COMPLETE_MSG, tylko opcje raportu w segmencie *pierwszy* są honorowane przez menedżer kolejek.
 - Jeśli segmenty są pobierane jeden po jednym, a większość z nich ma jedną z opcji MQRO_COD_*, ale co najmniej jeden segment nie, nie można użyć opcji MQGMO_COMPLETE_MSG do pobrania komunikatów raportu przy użyciu pojedynczego wywołania MQGET lub użyć opcji MQGMO_ALL_SEGMENTS_AVAILABLE w celu wykrycia, kiedy wszystkie komunikaty raportu zostały odebrane.
4. W sieci MQ menedżery kolejek mogą mieć różne możliwości. Jeśli komunikat raportu dla segmentu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji, menedżer kolejek lub agent MCA domyślnie nie zawierają informacji o segmentach niezbędnych w komunikacie raportu. Może to utrudnić zidentyfikowanie oryginalnego komunikatu, który spowodował wygenerowanie raportu. Unikaj tej trudności, żądając danych za pomocą komunikatu raportu, tj. poprzez określenie odpowiednich opcji MQRO_*_WITH_DATA lub MQRO_*_WITH_FULL_DATA. Należy jednak pamiętać, że jeśli zostanie podana wartość MQRO_*_WITH_DATA, *mniej niż 100 bajtów* danych komunikatu aplikacji może zostać zwróconych do aplikacji, która pobiera komunikat raportu, jeśli komunikat raportu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji.

Treść deskryptora komunikatu dla komunikatu raportu: Gdy menedżer kolejek lub agent kanału komunikatów (MCA) generuje komunikat raportu, ustawia pola w deskrypcorze komunikatu na następujące wartości, a następnie umieszcza komunikat w normalny sposób.

Pole w strukturze MQMD	Użyta wartość
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	MQRO_NONE
<i>MsgType</i>	Raport_menedżera_mQMT
<i>Expiry</i>	MQEI_UNLIMITED
<i>Feedback</i>	W zależności od rodzaju raportu (MQFB_COA, MQFB_COD, MQFB_EXPIRATION lub MQRC_*)
<i>Encoding</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>CodedCharSetId</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>Format</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>Priority</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>Persistence</i>	Skopiowano z oryginalnego deskryptora komunikatu

Pole w strukturze MQMD	Użyta wartość
<i>MsgId</i>	Jak określono w opcjach raportu w oryginalnym deskrypcorze komunikatu
<i>CorrelId</i>	Jak określono w opcjach raportu w oryginalnym deskrypcorze komunikatu
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Puste
<i>ReplyToQMGr</i>	Nazwa menedżera kolejek.
<i>UserIdentifier</i>	Zgodnie z ustawioną opcją MQPMO_PASS_IDENTITY_CONTEXT
<i>AccountingToken</i>	Zgodnie z ustawioną opcją MQPMO_PASS_IDENTITY_CONTEXT
<i>ApplIdentityData</i>	Zgodnie z ustawioną opcją MQPMO_PASS_IDENTITY_CONTEXT
<i>PutApplType</i>	MQAT_QMGR lub, jeśli jest to właściwe dla agenta kanału komunikatów
<i>PutApplName</i>	Pierwsze 28 bajtów nazwy menedżera kolejek lub nazwy agenta kanału komunikatów. W przypadku komunikatów raportu wygenerowanych przez most IMS pole to zawiera nazwę grupy XCF i nazwę elementu XCF systemu IMS , do którego odnosi się komunikat.
<i>PutDate</i>	Data wysłania komunikatu raportu
<i>PutTime</i>	Czas wysłania komunikatu raportu
<i>ApplOriginData</i>	Puste
<i>GroupId</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MsgSeqNumber</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>Offset</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MsgFlags</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>OriginalLength</i>	Skopiowano z oryginalnego deskryptora komunikatu, jeśli nie ma wartości MQOL_UNDEFINED, i ustaw na długość oryginalnego komunikatu, w przeciwnym razie

Zaleca się, aby aplikacja generująca raport ustawiała podobne wartości, z wyjątkiem następujących:

- Pole *ReplyToQMGr* można ustawić na odstęp (menedżer kolejek zmienia to na nazwę lokalnego menedżera kolejek po umieszczeniu w nim komunikacie).
- Ustaw pola kontekstu przy użyciu opcji, która została użyta dla odpowiedzi, normalnie MQPMO_PASS_IDENTITY_CONTEXT.

Analizowanie pola raportu: pole *Report* zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu zażądał konkretnego raportu, muszą używać jednej z technik opisanych w sekcji [“Analizowanie pola raportu” na stronie 916](#).

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQRO_NONE.

StrucId (MQCHAR4)

Jest to identyfikator struktury, który musi być następujący:

MQMD_STRUC_ID

Identyfikator struktury deskryptora komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQMD_STRUC_ID_ARRAY; ma taką samą wartość jak MQMD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQMD_STRUC_ID.

UserIdentifier (MQCHAR12)

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [“Przegląd deskryptora MQMD”](#) na stronie 418 i [Kontekst komunikatu](#).

UserIdentifier określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku.

Po odebraniu komunikatu należy użyć opcji *UserIdentifier* w polu *AlternateUserId* parametru *ObjDesc* kolejnych wywołań MQOPEN lub MQPUT1, aby wykonać sprawdzenie autoryzacji dla użytkownika produktu *UserIdentifier* zamiast aplikacji wykonujących otwarcie.

Gdy menedżer kolejek generuje te informacje dla wywołania MQPUT lub MQPUT1, wykonaj następujące czynności:

- W systemie z/OS menedżer kolejek używa *AlternateUserId* z parametru *ObjDesc* wywołania MQOPEN lub MQPUT1, jeśli została określona opcja MQOO_ALTERNATE_USER_AUTHORITY lub MQPMO_ALTERNATE_USER_AUTHORITY. Jeśli odpowiednia opcja nie została określona, menedżer kolejek używa identyfikatora użytkownika określonego w środowisku.
- W innych środowiskach menedżer kolejek zawsze używa identyfikatora użytkownika określonego w środowisku.

Gdy identyfikator użytkownika jest określany na podstawie środowiska:

- W systemie z/OS menedżer kolejek używa:
 - Dla systemu MVS (zadania wsadowego) identyfikator użytkownika z karty zadania JES lub uruchomionego zadania
 - Dla TSO identyfikator użytkownika propagowany do zadania podczas wprowadzania zadania
 - W przypadku systemu CICS identyfikator użytkownika powiązany z zadaniem
 - W przypadku systemu IMS identyfikator użytkownika zależy od typu aplikacji:
 - Przez:
 - Regiony BMP niezwiązane z komunikatem
 - Niekomunikat regionów IFP
 - Komunikat BMP i regiony IFP komunikatu, które *nie* wydały pomyślnego wywołania GU
- Menedżer kolejek używa identyfikatora użytkownika z karty JES regionu JES lub identyfikatora użytkownika TSO. Jeśli są one puste lub mają wartość NULL, używa nazwy bloku specyfikacji programu (PSB).
- Przez:
 - Komunikaty BMP i regiony IFP komunikatu, które *mają*, wydały pomyślne wywołanie GU
 - Regiony MPP
- menedżer kolejek używa jednego z następujących elementów:
- Identyfikator zalogowanego użytkownika powiązany z komunikatem
 - Nazwa terminalu logicznego (LTERM)
 - Identyfikator użytkownika z karty pracy regionu JES
 - Identyfikator użytkownika TSO
 - Nazwa PSB

- W systemie IBM imenedżer kolejek używa nazwy profilu użytkownika powiązanego z zadaniem aplikacji.
- W systemie Systemy UNIX menedżer kolejek używa:

- Nazwa logowania aplikacji
- Efektywny identyfikator użytkownika procesu, jeśli logowanie nie jest dostępne
- Identyfikator użytkownika powiązany z transakcją, jeśli aplikacja jest transakcją produktu CICS .
- W systemach Windows menedżer kolejek korzysta z pierwszych 12 znaków nazwy zalogowanego użytkownika.

To pole jest zwykle polem wyjściowym wygenerowanym przez menedżer kolejek, ale dla wywołania MQPUT lub MQPUT1 można ustawić to pole jako pole wejściowe/wyjściowe i określić pole *UserIdentification* , a nie pozwolić menedżerowi kolejek generować te informacje. Określ wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT w parametrze *PutMsgOpts* i podaj identyfikator użytkownika w polu *UserIdentifier* , jeśli nie chcesz, aby menedżer kolejek generował pole *UserIdentifier* dla wywołania MQPUT lub MQPUT1 .

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT został określony w parametrze *PutMsgOpts* . Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące znaki w puste znaki. Jeśli parametr MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT nie zostanie określony, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *UserIdentifier* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *UserIdentifier* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis opcji MQPMO_RETAIN, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest używana jako *UserIdentifier* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *UserIdentifier* we wszystkich publikacjach wysyłanych do tych subskrybentów. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C, a 12 pustych znaków w innych językach programowania.

Wersja (MQLONG)

Jest to numer wersji struktury i musi być jedną z następujących wartości:

MQMD_VERSION_1

Struktura deskryptora komunikatu Version-1 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQMD_VERSION_2

Struktura deskryptora komunikatu Version-2 .

Ta wersja jest obsługiwana we wszystkich środowiskach IBM MQ V6.0 i nowszych oraz w produkcji IBM MQ MQI clients połączonym z tymi systemami.

Uwaga: Jeśli używany jest deskryptor MQMD w wersji version-2 , menedżer kolejek wykonuje dodatkowe sprawdzenia wszystkich struktur nagłówka produktu MQ , które mogą być obecne na początku danych komunikatu aplikacji. Szczegółowe informacje na ten temat zawiera uwagi dotyczące składni wywołania MQPUT.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MQMD_CURRENT_VERSION

Bieżąca wersja struktury deskryptora komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQMD_VERSION_1.

Wartości początkowe i deklaracje języków dla deskryptora MQMD

Tabela 55. Początkowe wartości pól w MQMD dla deskryptora MQMD

Nazwa pola	Nazwa stałej	Wartość stałej
<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	Zależy od środowiska
<i>CodedCharSetId</i>	MQCCSI_Q_MGR	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Priority</i>	MQPRI_PRIORITY_AS_Q_DEF	-1
<i>Persistence</i>	MQPER_PERSISTENCE_AS_Q_DEF	2
<i>MsgId</i>	MQMI_NONE	Wartości null
<i>CorrelId</i>	MQCI_NONE	Wartości null
<i>BackoutCount</i>	Brak	0
<i>ReplyToQ</i>	Brak	Pusty łańcuch lub odstępy
<i>ReplyToQMgr</i>	Brak	Pusty łańcuch lub odstępy
<i>UserIdentifier</i>	Brak	Pusty łańcuch lub odstępy
<i>AccountingToken</i>	MQACT_NONE	Wartości null
<i>ApplIdentityData</i>	Brak	Pusty łańcuch lub odstępy
<i>PutApplType</i>	MQAT_NO_CONTEXT	0
<i>PutApplName</i>	Brak	Pusty łańcuch lub odstępy
<i>PutDate</i>	Brak	Pusty łańcuch lub odstępy
<i>PutTime</i>	Brak	Pusty łańcuch lub odstępy
<i>ApplOriginData</i>	Brak	Pusty łańcuch lub odstępy
<i>GroupId</i>	MQGI_NONE	Wartości null
<i>MsgSeqNumber</i>	Brak	1
<i>Offset</i>	Brak	0
<i>MsgFlags</i>	MQMF_NONE	0
<i>OriginalLength</i>	MQOL_NIEZDEFINIOWANY	-1

Tabela 55. Początkowe wartości pól w MQMD dla deskryptora MQMD (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: 1. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. 2. W języku programowania C: zmienna makraMQMD_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze: MQMD MyMD = {MQMD_DEFAULT};		

Deklaracja C

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    Report;           /* Options for report messages */
    MQLONG    MsgType;          /* Message type */
    MQLONG    Expiry;           /* Message lifetime */
    MQLONG    Feedback;         /* Feedback or reason code */
    MQLONG    Encoding;         /* Numeric encoding of message data */
    MQLONG    CodedCharSetId;   /* Character set identifier of message
                                data */
    MQCHAR8   Format;           /* Format name of message data */
    MQLONG    Priority;          /* Message priority */
    MQLONG    Persistence;      /* Message persistence */
    MQBYTE24  MsgId;           /* Message identifier */
    MQBYTE24  CorrelId;         /* Correlation identifier */
    MQLONG    BackoutCount;     /* Backout counter */
    MQCHAR48  ReplyToQ;         /* Name of reply queue */
    MQCHAR48  ReplyToQMGr;      /* Name of reply queue manager */
    MQCHAR12  UserIdentifier;    /* User identifier */
    MQBYTE32  AccountingToken;  /* Accounting token */
    MQCHAR32  ApplIdentityData; /* Application data relating to
                                identity */
    MQLONG    PutApplType;      /* Type of application that put the
                                message */
    MQCHAR28  PutApplName;      /* Name of application that put the
                                message */
    MQCHAR8   PutDate;          /* Date when message was put */
    MQCHAR8   PutTime;          /* Time when message was put */
    MQCHAR4   ApplOriginData;   /* Application data relating to origin */
    MQBYTE24  GroupId;          /* Group identifier */
    MQLONG    MsgSeqNumber;     /* Sequence number of logical message
                                within group */
    MQLONG    Offset;           /* Offset of data in physical message
                                from start of logical message */
    MQLONG    MsgFlags;         /* Message flags */
    MQLONG    OriginalLength;   /* Length of original message */
};
```

Deklaracja języka COBOL

```
**      MQMD structure
10 MQMD.
**      Structure identifier
15 MQMD-STRUCID      PIC X(4).
**      Structure version number
15 MQMD-VERSION      PIC S9(9) BINARY.
**      Options for report messages
15 MQMD-REPORT       PIC S9(9) BINARY.
**      Message type
15 MQMD-MSGTYPE      PIC S9(9) BINARY.
**      Message lifetime
15 MQMD-EXPIRY       PIC S9(9) BINARY.
**      Feedback or reason code
15 MQMD-FEEDBACK     PIC S9(9) BINARY.
**      Numeric encoding of message data
```



```

15 MQMD-ENCODING          PIC S9(9) BINARY.
** Character set identifier of message data
15 MQMD-CODEDCHARSETID   PIC S9(9) BINARY.
** Format name of message data
15 MQMD-FORMAT           PIC X(8).
** Message priority
15 MQMD-PRIORITY         PIC S9(9) BINARY.
** Message persistence
15 MQMD-PERSISTENCE      PIC S9(9) BINARY.
** Message identifier
15 MQMD-MSGID            PIC X(24).
** Correlation identifier
15 MQMD-CORRELID         PIC X(24).
** Backout counter
15 MQMD-BACKOUTCOUNT    PIC S9(9) BINARY.
** Name of reply queue
15 MQMD-REPLYTOQ         PIC X(48).
** Name of reply queue manager
15 MQMD-REPLYTOQMGR      PIC X(48).
** User identifier
15 MQMD-USERIDENTIFIER   PIC X(12).
** Accounting token
15 MQMD-ACCOUNTINGTOKEN  PIC X(32).
** Application data relating to identity
15 MQMD-APPLIDENTITYDATA PIC X(32).
** Type of application that put the message
15 MQMD-PUTAPPLTYPE      PIC S9(9) BINARY.
** Name of application that put the message
15 MQMD-PUTAPPLNAME      PIC X(28).
** Date when message was put
15 MQMD-PUTDATE          PIC X(8).
** Time when message was put
15 MQMD-PUTTIME          PIC X(8).
** Application data relating to origin
15 MQMD-APPLORIGINDATA   PIC X(4).
** Group identifier
15 MQMD-GROUPID          PIC X(24).
** Sequence number of logical message within group
15 MQMD-MSGSEQUENumber   PIC S9(9) BINARY.
** Offset of data in physical message from start of logical message
15 MQMD-OFFSET           PIC S9(9) BINARY.
** Message flags
15 MQMD-MSGFLAGS         PIC S9(9) BINARY.
** Length of original message
15 MQMD-ORIGINALLENGTH   PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQMD based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 Report           fixed bin(31),    /* Options for report messages */
3 MsgType          fixed bin(31),    /* Message type */
3 Expiry           fixed bin(31),    /* Message lifetime */
3 Feedback         fixed bin(31),    /* Feedback or reason code */
3 Encoding         fixed bin(31),    /* Numeric encoding of message
                                     data */
3 CodedCharSetId   fixed bin(31),    /* Character set identifier of
                                     message data */
3 Format            char(8),          /* Format name of message data */
3 Priority          fixed bin(31),    /* Message priority */
3 Persistence      fixed bin(31),    /* Message persistence */
3 MsgId            char(24),         /* Message identifier */
3 CorrelId         char(24),         /* Correlation identifier */
3 BackoutCount     fixed bin(31),    /* Backout counter */
3 ReplyToQ         char(48),         /* Name of reply queue */
3 ReplyToQMgr      char(48),         /* Name of reply queue manager */
3 UserIdentifier   char(12),         /* User identifier */
3 AccountingToken  char(32),         /* Accounting token */
3 ApplIdentityData char(32),         /* Application data relating to
                                     identity */
3 PutApplType      fixed bin(31),    /* Type of application that put the
                                     message */
3 PutApplName      char(28),         /* Name of application that put the
                                     message */
3 PutDate          char(8),          /* Date when message was put */
3 PutTime          char(8),          /* Time when message was put */
3 ApplOriginData   char(4),          /* Application data relating to

```

```

                                origin */
3 GroupId          char(24),      /* Group identifier */
3 MsgSeqNumber     fixed bin(31), /* Sequence number of logical
                                message within group */
3 Offset          fixed bin(31), /* Offset of data in physical
                                message from start of logical
                                message */
3 MsgFlags        fixed bin(31), /* Message flags */
3 OriginalLength  fixed bin(31); /* Length of original message */

```

Deklaracja High Level Assembler

```

MQMD                                DSECT
MQMD_STRUCID                       DS   CL4   Structure identifier
MQMD_VERSION                       DS   F     Structure version number
MQMD_REPORT                        DS   F     Options for report messages
MQMD_MSGTYPE                       DS   F     Message type
MQMD_EXPIRY                        DS   F     Message lifetime
MQMD_FEEDBACK                      DS   F     Feedback or reason code
MQMD_ENCODING                      DS   F     Numeric encoding of message data
MQMD_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)

```

Wizualna deklaracja podstawowa

```

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'
UserIdentifer 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'

```

PutApplName	As String*28	'message' '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-Rozszerzenie deskryptora komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 56. Pola w MQMDE		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQMDE	StrucLength
<i>Encoding</i>	Kodowanie numeryczne danych, które są następujące: MQMDE	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków dla danych, które są następujące: MQMDE	CodedCharSetId
<i>Format</i>	Nazwa formatu danych, które są następujące: MQMDE	Formatowanie
<i>Flags</i>	Flagi ogólne	Flagi
<i>GroupId</i>	Identyfikator grupy	GroupId
<i>MsgSeqNumber</i>	Numer kolejny komunikatu logicznego w grupie	Numer_kolejny_komunikatu
<i>Offset</i>	Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego	Depozycja
<i>MsgFlags</i>	Flagi komunikatu	MsgFlags
<i>OriginalLength</i>	Długość oryginalnego komunikatu	OriginalLength

Przegląd produktu MQMDE

Dostępność: Wszystkie systemy IBM MQ oraz klienci IBM MQ połączone z tymi systemami.

Cel: Struktura MQMDE opisuje dane, które czasami występują przed danymi komunikatu aplikacji. Struktura zawiera te pola MQMD, które istnieją w MQMD w wersji version-2, ale nie w przypadku deskryptora MQMD w wersji version-1.

Nazwa formatu: MQFMT_MD_EXTENSION.

Zestaw znaków i kodowanie: Dane w produkcie MQMDE muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Te dane są nadawane przez atrybut menedżera kolejek produktu *CodedCharSetId* i wartość MQENC_NATIVE dla języka programowania C.

Ustaw zestaw znaków i kodowanie wartości MQMDE w polach *CodedCharSetId* i *Encoding* w:

- MQMD (jeśli struktura MQMDE znajduje się na początku danych komunikatu), lub

- Struktura nagłówka, która poprzedza strukturę MQMDE (wszystkie inne obserwacje).

Jeśli tabela MQMDE nie znajduje się w zestawie znaków i kodowaniu menedżera kolejek, wartość MQMDE jest akceptowana, ale nie została uhonorowana, oznacza to, że MQMDE jest traktowane jako dane komunikatu.

Uwaga: W produkcie Windows aplikacje skompilowane przy użyciu programu Micro Focus COBOL korzystają z wartości MQENC_NATIVE, która różni się od kodowania menedżera kolejek. Mimo że pola liczbowe w strukturze MQMD w wywołaniach MQPUT, MQPUT1 i MQGET muszą znajdować się w kodowaniu Micro Focus COBOL, pola liczbowe w strukturze MQMDE muszą znajdować się w kodowaniu menedżera kolejek. Ten ostatni jest nadawany przez MQENC_NATIVE dla języka programowania C i ma wartość 546.

Użycie: Aplikacje, które używają deskryptora MQMD w wersji version-2, nie będą napotkały struktury MQMDE. Jednak wyspecjalizowane aplikacje i aplikacje, które w dalszym ciągu używają deskryptora MQMD z wersji version-1, mogą napotkać w niektórych sytuacjach wywołanie MQMDE. Struktura MQMDE może wystąpić w następujących okolicznościach:

- Określone w wywołaniach MQPUT i MQPUT1
- Zwrócone przez wywołanie MQGET
- W komunikatach w kolejkach transmisji

MQMDE określony w wywołaniach MQPUT i MQPUT1: w wywołaniach MQPUT i MQPUT1, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, aplikacja może opcjonalnie prefiksować dane komunikatu za pomocą MQMDE, ustawiając pole *Format* w strukturze MQMD na MQFMT_MD_EXTENSION, aby wskazać, że jest obecny produkt MQMDE. Jeśli aplikacja nie udostępnia wywołania MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w MQMDE. Wartości domyślne używane przez menedżera kolejek są takie same, jak początkowe wartości struktury; patrz Tabela 58 na stronie 475.

If the application provides a version-2 MQMD *oraz* prefixes the application message data with an MQMDE, the structures are processed as shown in Tabela 57 na stronie 472.

Tabela 57. Działanie menedżera kolejek, gdy MQMDE jest określone w MQPUT lub MQPUT1 dla MQMDE			
Wersja MQMD	Wartości w polach version-2	Wartości odpowiednich pól w MQMDE	Działanie podjęte przez menedżera kolejek
1	-	Ważne	MQMDE jest honorowany
2	Domyślny	Ważne	MQMDE jest honorowany
2	Niedomyślna	Ważne	MQMDE jest traktowane jako dane komunikatu
1 lub 2	Dowolna	Niepoprawne	Wywołanie nie powiodło się z odpowiednim kodem przyczyny
1 lub 2	Dowolna	MQMDE znajduje się w niewłaściwym zestawie znaków lub kodowaniu, albo jest nieobsługiwana wersją	MQMDE jest traktowane jako dane komunikatu
Uwaga: Jeśli w systemie z/OS aplikacja określa version-1 MQMD z MQMDE, menedżer kolejek sprawdza poprawność MQMDE tylko wtedy, gdy w kolejce znajduje się <i>IndexType</i> o wartości MQIT_GROUP_ID.			

Jest jedna szczególna sprawa. Jeśli aplikacja używa deskryptora MQMD z wersji version-2 w celu umieszczenia komunikatu, który jest segmentem (to znaczy, że ustawiona jest flaga MQMF_SEGMENT lub MQMF_LAST_SEGMENT), a nazwa formatu w strukturze MQMD to MQFMT_DEAD_LETTER_HEADER, menedżer kolejek generuje strukturę MQMDE i wstawia ją między strukturą MQDLH a danymi, które są

zgodne z tą nazwą. W deskryptywie MQMD, który menedżer kolejek zachowuje z komunikatem, pola version-2 są ustawiane na wartości domyślne.

Kilka pól istniejących w strukturze MQMD version-2, ale nie version-1 MQMD, są polami wejściowymi/wyjściowymi w tabelach MQPUT i MQPUT1. Jednak menedżer kolejek *nie* zwraca żadnych wartości w równoważnych polach w MQMDE na danych wyjściowych z wywołań MQPUT i MQPUT1. Jeśli aplikacja wymaga tych wartości wyjściowych, musi ona używać deskryptora MQMD z wersji version-2.

MQMDE zwrócone przez wywołanie MQGET: W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, menedżer kolejek prefiksuje komunikat zwrócony z MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość niedomyślną. Menedżer kolejek ustawia pole *Format* w deskryptywie MQMD na wartość MQFMT_MD_EXTENSION, aby wskazać, że jest obecna MQMDE.

Jeśli aplikacja udostępnia parametr MQMDE na początku parametru *Buffer*, MQMDE jest ignorowane. W przypadku powrotu z wywołania MQGET jest on zastępowany przez MQMDE dla komunikatu (jeśli jest wymagany) lub nadpisywany przez dane komunikatu aplikacji (jeśli MQMDE nie jest potrzebne).

Jeśli wywołanie MQMDE zwróci wartość MQMDE, dane w MQMDE są zwykle w zestawie znaków i kodowaniu menedżera kolejek. Jednak MQMDE może znajdować się w innym zestawie znaków i kodowaniu, jeśli:

- Produkt MQMDE został potraktowany jako dane w wywołaniu MQPUT lub MQPUT1 (patrz [Tabela 57 na stronie 472](#) w celu uzyskania informacji o okolicznościach, które mogą być przyczyną tego wywołania).
- Komunikat został odebrany ze zdalnego menedżera kolejek połączonego przez połączenie TCP, a odbierający agent kanału komunikatów (MCA) nie został skonfigurowany poprawnie.

Uwaga: W produkcie Windows aplikacje skompilowane przy użyciu programu Micro Focus COBOL korzystają z wartości MQENC_NATIVE, która różni się od kodowania menedżera kolejek (patrz wyżej).

MQMDE w komunikatach w kolejkach transmisji: komunikaty w kolejkach transmisji są poprzedzane strukturą MQXQH, która zawiera w sobie deskryptor MQMD programu version-1. Może być również obecny produkt MQMDE, który znajduje się między strukturą MQXQH a danymi komunikatu aplikacji, ale jest on zwykle obecny tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość niedomyślną.

Między strukturą MQXQH a danymi komunikatu aplikacji mogą występować również inne struktury nagłówka produktu MQ. Na przykład, gdy występuje nagłówek niedostarczonych komunikatów MQDLH, a komunikat nie jest segmentem, to kolejność jest następująca:

- MQXQH (zawiera element MQMD w wersji version-1)
- MQMDE
- MQDLH
- dane komunikatu aplikacji

Pola dla MQMDE

Struktura MQMDE zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CodedCharSetId (MQLONG)

Określa identyfikator zestawu znaków dla danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych znakowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie można użyć tabeli MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcyjce MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz klienty IBM MQ połączone z tymi systemami.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Kodowanie (MQLONG)

Określa kodowanie liczbowe dla danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych liczbowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Więcej informacji na temat kodowania danych można znaleźć w sekcji *Encoding* opisanej w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#).

Wartością początkową tego pola jest MQENC_NATIVE.

Flagi (MQLONG)

Można określić następującą opcję:

MQMDEF_NONE

Brak flag.

Wartością początkową tego pola jest MQMDEF_NONE.

Format (MQCHAR8)

Określa nazwę formatu danych, które są zgodne ze strukturą MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Więcej informacji na temat nazw formatów znajduje się w sekcji *Format* opisanej w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#).

Wartością początkową tego pola jest MQFMT_NONE.

GroupId (MQBYTE24)

Zapoznaj się z polem *GroupId*, które opisano w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#). Wartością początkową tego pola jest MQGI_NONE.

MsgFlags (MQLONG)

Zapoznaj się z polem *MsgFlags*, które opisano w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#). Wartością początkową tego pola jest MQMF_NONE.

Liczba MsgSeq (MQLONG)

Zapoznaj się z polem *MsgSeqNumber*, które opisano w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#). Wartością początkową tego pola jest 1.

Przesunięcie (MQLONG)

Zapoznaj się z polem *Offset*, które opisano w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#). Wartością początkową tego pola jest 0.

OriginalLength (MQLONG)

Zapoznaj się z polem *OriginalLength*, które opisano w sekcji [“MQMD-deskryptor komunikatu” na stronie 417](#). Wartością początkową tego pola jest MQOL_UNDEFINED.

StrucId (MQCHAR4)

Wartość musi być następująca:

MQMDE_STRUC_ID

Identyfikator struktury rozszerzenia deskryptora komunikatu.

Dla języka programowania w języku C jest również zdefiniowana stała MQMDE_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQMDE_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQMDE_STRUC_ID.

StrucLength (MQLONG)

Jest to długość struktury MQMDE. Zdefiniowana jest następująca wartość:

MQMDE_LENGTH_2

Długość struktury rozszerzenia deskryptora komunikatu version-2 .

Początkowa wartość tego pola to MQMDE_LENGTH_2.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQMDE_VERSION_2

Struktura rozszerzenia deskryptora komunikatu Version-2 .

Następująca stała określa numer wersji bieżącej wersji:

MQMDE_CURRENT_VERSION

Bieżąca wersja struktury rozszerzenia deskryptora komunikatu.

Początkowa wartość tego pola to MQMDE_VERSION_2.

Wartości początkowe i deklaracje języków dla produktu MQMDE

Tabela 58. Początkowe wartości pól w MQMDE dla MQMDE		
Nazwa pola	Nazwa stałej	Wartość stałej
<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	Zależy od środowiska
<i>CodedCharSetId</i>	MQCCSI_UNDEFINED	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Flags</i>	MQMDEF_NONE	0
<i>GroupId</i>	MQGI_NONE	Wartości null
<i>MsgSeqNumber</i>	Brak	1
<i>Offset</i>	Brak	0
<i>MsgFlags</i>	MQMF_NONE	0
<i>OriginalLength</i>	MQOL_NIEZDEFINIOWANY	-1
Uwagi:		
1. Symbol ↵ reprezentuje pojedynczy pusty znak.		
2. W języku programowania C: zmienna makraMQMDE_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQMDE MyMDE = {MQMDE_DEFAULT};</pre>		

Deklaracja C

```

typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    StrucLength;       /* Length of MQMDE structure */
    MQLONG    Encoding;          /* Numeric encoding of data that follows
                                MQMDE */
    MQLONG    CodedCharSetId;    /* Character-set identifier of data that
                                follows MQMDE */
    MQCHAR8   Format;            /* Format name of data that follows
                                MQMDE */
    MQLONG    Flags;             /* General flags */
    MQBYTE24  GroupId;           /* Group identifier */
    MQLONG    MsgSeqNumber;      /* Sequence number of logical message
                                within group */
    MQLONG    Offset;            /* Offset of data in physical message from
                                start of logical message */
    MQLONG    MsgFlags;          /* Message flags */
    MQLONG    OriginalLength;    /* Length of original message */
};

```

Deklaracja języka COBOL

```

**      MQMDE structure
10 MQMDE.
**      Structure identifier
15 MQMDE-STRUCID          PIC X(4).
**      Structure version number
15 MQMDE-VERSION          PIC S9(9) BINARY.
**      Length of MQMDE structure
15 MQMDE-STRUCLength     PIC S9(9) BINARY.
**      Numeric encoding of data that follows MQMDE
15 MQMDE-ENCODING         PIC S9(9) BINARY.
**      Character-set identifier of data that follows MQMDE
15 MQMDE-CODEDCHARSETID PIC S9(9) BINARY.
**      Format name of data that follows MQMDE
15 MQMDE-FORMAT           PIC X(8).
**      General flags
15 MQMDE-FLAGS            PIC S9(9) BINARY.
**      Group identifier
15 MQMDE-GROUPID          PIC X(24).
**      Sequence number of logical message within group
15 MQMDE-MSGSEQNUMBER     PIC S9(9) BINARY.
**      Offset of data in physical message from start of logical message
15 MQMDE-OFFSET           PIC S9(9) BINARY.
**      Message flags
15 MQMDE-MSGFLAGS         PIC S9(9) BINARY.
**      Length of original message
15 MQMDE-ORIGINALLENGTH PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQMDE based,
3 StrucId          char(4),           /* Structure identifier */
3 Version          fixed bin(31),     /* Structure version number */
3 StrucLength      fixed bin(31),     /* Length of MQMDE structure */
3 Encoding         fixed bin(31),     /* Numeric encoding of data that
                                follows MQMDE */
3 CodedCharSetId   fixed bin(31),     /* Character-set identifier of data
                                that follows MQMDE */
3 Format           char(8),            /* Format name of data that follows
                                MQMDE */
3 Flags           fixed bin(31),     /* General flags */
3 GroupId         char(24),           /* Group identifier */
3 MsgSeqNumber     fixed bin(31),     /* Sequence number of logical message
                                within group */
3 Offset          fixed bin(31),     /* Offset of data in physical message
                                from start of logical message */
3 MsgFlags        fixed bin(31),     /* Message flags */
3 OriginalLength   fixed bin(31);    /* Length of original message */

```

Deklaracja High Level Assembler


```

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
MQMDE_CODEDCHARSETID DS    F     Character-set identifier of data that
*              follows MQMDE
MQMDE_FORMAT    DS    CL8   Format name of data that follows MQMDE
MQMDE_FLAGS     DS    F     General flags
MQMDE_GROUPID   DS    XL24  Group identifier
MQMDE_MSGSEQNUMBER DS    F     Sequence number of logical message
*              within group
MQMDE_OFFSET    DS    F     Offset of data in physical message from
*              start of logical message
MQMDE_MSGFLAGS  DS    F     Message flags
MQMDE_ORIGINALLENGTH DS    F     Length of original message
*
MQMDE_LENGTH    EQU    *-MQMDE
                ORG    MQMDE
MQMDE_AREA      DS      CL(MQMDE_LENGTH)

```

Wizualna deklaracja podstawowa

```

Type MQMDE
  StrucId      As String*4 'Structure identifier'
  Version      As Long      'Structure version number'
  StrucLength  As Long      'Length of MQMDE structure'
  Encoding     As Long      'Numeric encoding of data that follows'
                  'MQMDE'
  CodedCharSetId As Long      'Character-set identifier of data that'
                  'follows MQMDE'
  Format       As String*8   'Format name of data that follows MQMDE'
  Flags        As Long      'General flags'
  GroupId      As MQBYTE24  'Group identifier'
  MsgSeqNumber As Long      'Sequence number of logical message within'
                  'group'
  Offset       As Long      'Offset of data in physical message from'
                  'start of logical message'
  MsgFlags     As Long      'Message flags'
  OriginalLength As Long      'Length of original message'
End Type

```

MQMHBO-uchwyt komunikatu do opcji buforu

W poniższej tabeli podsumowano pola w strukturze. Struktura MQMHBO-uchwyt komunikatu do opcji buforu

Tabela 59. Pola w MQMHBO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem MQMHBUF	Opcje

Przegląd produktu MQMHBO

Dostępność: Wszystkie systemy IBM MQ i IBM MQ MQI clients.

Przeznaczenie: Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki bufor są generowane z uchwytów komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQMHBUF.

Zestaw znaków i kodowanie: Dane w tabeli MQMHBO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla MQMHBO

Uchwyt komunikatu do struktury opcji buforu-pola

Struktura MQMHBO zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**:

Opcje (MQLONG)

Uchwyt komunikatu do struktury opcji buforu-pole Opcje

Te opcje sterują działaniem MQMHBUF.

Należy podać następującą opcję:

MQMHBO_PROPERTIES_IN_MQRFH2

Przekształcając właściwości z uchwytu komunikatu w bufor, przekształć je w format MQRFH2 .

Opcjonalnie można również określić następującą opcję. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

MQMHBO_DELETE_PROPERTIES

Właściwości, które są dodawane do buforu, są usuwane z uchwytu komunikatu. Jeśli wywołanie nie powiedzie się, żadne właściwości nie zostaną usunięte.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQMHBO_PROPERTIES_IN_MQRFH2.

StrucId (MQCHAR4)

Uchwyt komunikatu do struktury opcji buforu-pole StrucId

Jest to identyfikator struktury. Wartość musi być następująca:

MQMHBO_STRUC_ID

Identyfikator uchwytu komunikatu do struktury opcji buforu.

Dla języka programowania C zdefiniowana jest również stała MQMHBO_STRUC_ID_ARRAY; ma taką samą wartość jak MQMHBO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQMHBO_STRUC_ID.

Wersja (MQLONG)

Uchwyt komunikatu do struktury opcji buforu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQMHBO_VERSION_1

Numer wersji dla uchwytu komunikatu do struktury opcji buforu.

Następująca stała określa numer wersji bieżącej wersji:

MQMHBO_CURRENT_VERSION

Bieżąca wersja uchwytu komunikatu do struktury opcji buforu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQMHBO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQMHBO

Uchwyt komunikatu do struktury buforu-wartości początkowe

Tabela 60. Początkowe wartości pól w MQMHBO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQMHBO_STRUC_ID	' MHBO '
<i>Version</i>	MQMHBO_VERSION_1	1
<i>Options</i>	MQMHBO_PROPERTIES_IN_MQRFH2	

Tabela 60. Początkowe wartości pól w MQMHBO (kontynuacja)		
Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: - Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. - W języku programowania C: zmienna makraMQMHBO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQMHBO MyMHBO = {MQMHBO_DEFAULT};</pre>		

Deklaracja C

Uchwyt komunikatu do struktury opcji buforu-deklaracja języka C

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
    MQCHAR4  StrucId;          /* Structure identifier */
    MQLONG   Version;         /* Structure version number */
    MQLONG   Options;         /* Options that control the action of
                               MQMHBUF */
};
```

Deklaracja języka COBOL

Uchwyt komunikatu do struktury opcji buforu-deklaracja języka COBOL

```
** MQMHBO structure
10 MQMHBO.
**   Structure identifier
15 MQMHBO-STRUCID          PIC X(4).
**   Structure version number
15 MQMHBO-VERSION         PIC S9(9) BINARY.
**   Options that control the action of MQMHBUF
15 MQMHBO-OPTIONS        PIC S9(9) BINARY.
```

Deklaracja PL/I

Uchwyt komunikatu do struktury opcji buforu-deklaracja języka PL/I

```
Dcl
1 MQMHBO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options that control the action
                               of MQMHBUF */
```

Deklaracja High Level Assembler

Uchwyt komunikatu do struktury opcji buforu-Deklaracja w języku asemblera

```
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-deskryptor obiektu

W poniższej tabeli podsumowano pola w strukturze.

Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>ObjectType</i>	Typ obiektu	ObjectType
<i>ObjectName</i>	Nazwa obiektu	ObjectName
<i>ObjectQMgrName</i>	Nazwa menedżera kolejek obiektów	NazwaObjectQMgr
<i>DynamicQName</i>	Nazwa kolejki dynamicznej	DynamicQName
<i>AlternateUserId</i>	Alternatywny identyfikator użytkownika	IdentyfikatorAlternateUser
Uwaga: Pozostałe pola są ignorowane, jeśli parametr <i>Version</i> jest mniejszy niż MQOD_VERSION_2.		
<i>RecsPresent</i>	Liczba obecnych rekordów obiektów	RecsPresent
<i>KnownDestCount</i>	Liczba pomyślnie otwartych kolejek lokalnych	KnownDestLiczba
<i>UnknownDestCount</i>	Liczba pomyślnie otwartych kolejek zdalnych	UnknownDestLiczba
<i>InvalidDestCount</i>	Liczba kolejek, których otwarcie nie powiodło się	InvalidDestLiczba
<i>ObjectRecOffset</i>	Przesunięcie pierwszego rekordu obiektu od początku MQOD	ObjectRecPrzesunięcie
<i>ResponseRecOffset</i>	Przesunięcie pierwszego rekordu odpowiedzi od początku MQOD	PrzesunięcieResponseRec
<i>ObjectRecPtr</i>	Adres pierwszego rekordu obiektu	ObjectRecPtr
<i>ResponseRecPtr</i>	Adres pierwszego rekordu odpowiedzi	ResponseRecPtr
Uwaga: Pozostałe pola są ignorowane, jeśli parametr <i>Version</i> jest mniejszy niż MQOD_VERSION_3.		
<i>AlternateSecurityId</i>	Alternatywny identyfikator zabezpieczeń	IdentyfikatorAlternateSecurity
<i>ResolvedQName</i>	Rozstrzygnięta nazwa kolejki	ResolvedQName
<i>ResolvedQMgrName</i>	Rozstrzygnięta nazwa menedżera kolejek	ResolvedQMgrNazwa
Uwaga: Pozostałe pola są ignorowane, jeśli parametr <i>Version</i> jest mniejszy niż MQOD_VERSION_4.		
<i>ObjectString</i>	Długa nazwa obiektu	ObjectString
<i>SelectionString</i>	Łańcuch wyboru	SelectionString
<i>ResObjectString</i>	Rozstrzygnięta długa nazwa obiektu	ResObjectŁańcuch
<i>ResolvedType</i>	Rozstrzygnięty typ obiektu	ResolvedType

Przegląd produktu MQOD

Dostępność: Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQOD służy do określania obiektu według nazwy. Poprawne są następujące typy obiektów:

- Kolejka lub lista dystrybucyjna
- Lista nazw
- Definicja procesu
- Menedżer kolejek

- Temat

Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQOPEN i MQPUT1 .

Wersja: Bieżąca wersja programu MQOD to MQOD_VERSION_4. Aplikacje, które mają być używane do portu między kilkoma środowiskami, muszą mieć pewność, że wymagana wersja programu MQOD jest obsługiwana we wszystkich środowiskach, których dotyczy. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję programu MQOD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQOD_VERSION_1. Aby użyć pól, które nie są obecne w strukturze *version-1* , aplikacja musi ustawić pole *Version* na numer wersji wymaganej wersji.

Aby otworzyć listę dystrybucyjną, *Version* musi mieć wartość MQOD_VERSION_2 lub większą.

Zestaw znaków i kodowanie: Dane w tabeli MQOD muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQOD

Struktura MQOD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Identyfikator AlternateSecurity(MQBYTE40)

Jest to identyfikator zabezpieczeń przekazywany razem z produktem *AlternateUserId* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji. Produkt *AlternateSecurityId* jest używany tylko wtedy, gdy:

- MQOO_ALTERNATE_USER_AUTHORITY jest określone w wywołaniu MQOPEN, lub
- opcja MQPMO_ALTERNATE_USER_AUTHORITY jest określona w wywołaniu MQPUT1 ,

Pole *i* pole *AlternateUserId* nie jest całkowicie puste w stosunku do pierwszego znaku o kodzie zero lub do końca pola.

W systemie Windows można użyć *AlternateSecurityId* do podania identyfikatora zabezpieczeń (SID) produktu Windows , który jednoznacznie identyfikuje *AlternateUserId*. Identyfikator SID dla użytkownika można uzyskać z systemu Windows , korzystając z wywołania funkcji API *LookupAccountName()* Windows .

W systemie z/OS to pole jest ignorowane.

Pole *AlternateSecurityId* ma następującą strukturę:

- Pierwszy bajt jest binarną liczbą całkowitą zawierającą długość znaczących danych, które są następujące; wartość nie obejmuje samego bajtu o długości. Jeśli identyfikator zabezpieczeń nie jest obecny, długość wynosi zero.
- Drugi bajt wskazuje typ identyfikatora zabezpieczeń, który jest obecny. Możliwe są następujące wartości:

MQSIDT_NT_SECURITY_ID

Identyfikator zabezpieczeń produktu Windows .

MQSIDT_NONE

Brak identyfikatora zabezpieczeń.

- Trzeci i kolejne bajty aż do długości zdefiniowanej przez pierwszy bajt zawierają sam identyfikator zabezpieczeń.
- Pozostałe bajty w polu są ustawione na zero binarne.

Można użyć następującej wartości specjalnej:

MQSID_NONE

Nie określono identyfikatora zabezpieczeń.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQSID_NONE_ARRAY; ma ona taką samą wartość jak MQSID_NONE, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_SECURITY_ID_LENGTH. Wartością początkową tego pola jest MQSID_NONE. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_3.

Identyfikator AlternateUser(MQCHAR12)

Jeśli określono wartość MQOO_ALTERNATE_USER_AUTHORITY dla wywołania MQOPEN lub wartość MQPMO_ALTERNATE_USER_AUTHORITY dla wywołania MQPUT1, to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla otwartego, zamiast identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona. Niektóre operacje sprawdzania są jednak nadal wykonywane z bieżącym identyfikatorem użytkownika (na przykład sprawdzeniami kontekstowymi).

Jeśli określono wartość MQOO_ALTERNATE_USER_AUTHORITY lub MQPMO_ALTERNATE_USER_AUTHORITY, a pole to jest całkowicie puste, do pierwszego znaku o kodzie zero lub do końca pola, może to zakończyć się powodzeniem tylko wtedy, gdy nie jest wymagane uprawnienie użytkownika do otwarcia tego obiektu przy użyciu podanych opcji.

Jeśli nie określono ani parametru MQOO_ALTERNATE_USER_AUTHORITY, ani MQPMO_ALTERNATE_USER_AUTHORITY, to pole jest ignorowane.

W podanych środowiskach istnieją następujące różnice:

- W systemie z/OS do sprawdzania autoryzacji dla otwarcia używane są tylko pierwsze 8 znaków produktu *AlternateUserId*. Jednak bieżący identyfikator użytkownika musi być autoryzowany do określenia tego konkretnego alternatywnego identyfikatora użytkownika; dla tego sprawdzenia używane są wszystkie 12 znaków alternatywnego identyfikatora użytkownika. Identyfikator użytkownika musi zawierać tylko znaki dozwolone przez zewnętrznego menedżera zabezpieczeń.

Jeśli dla kolejki określono wartość *AlternateUserId*, to podczas umieszczania komunikatów wartość może być następnie używana przez menedżer kolejek. Jeśli opcje MQPMO_*_CONTEXT określone w wywołaniu MQPUT lub MQPUT1 powodują, że menedżer kolejek generuje informacje o kontekście tożsamości, menedżer kolejek umieszcza *AlternateUserId* w polu *UserIdentifier* w strukturze MQMD komunikatu, w miejsce bieżącego identyfikatora użytkownika.

- W innych środowiskach produkt *AlternateUserId* jest używany tylko do sprawdzania praw dostępu do otwieranego obiektu. Jeśli obiekt jest kolejką, produkt *AlternateUserId* nie ma wpływu na zawartość pola *UserIdentifier* w strukturze MQMD komunikatów wysyłanych przy użyciu tego uchwytu kolejki.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C, a 12 pustych znaków w innych językach programowania.

DynamicQName (MQCHAR48)

Jest to nazwa kolejki dynamicznej, która ma zostać utworzona przy użyciu wywołania MQOPEN. Ma to znaczenie tylko wtedy, gdy parametr *ObjectName* określa nazwę kolejki modelowej; we wszystkich pozostałych przypadkach *DynamicQName* jest ignorowany.

Znaki, które są poprawne w nazwie, są takie same jak w przypadku produktu *ObjectName*, z tą różnicą, że znak gwiazdki jest również poprawny. Nazwa, która jest pusta (lub jedna, w której występują tylko spacje przed pierwszym znakiem o kodzie zero), nie jest poprawna, jeśli *ObjectName* jest nazwą kolejki modelowej.

Jeśli ostatni niepusty znak w nazwie to gwiazdka (*), to menedżer kolejek zastępuje gwiazdkę łańcuchem znaków, który gwarantuje, że nazwa wygenerowana dla kolejki jest unikalna w lokalnym menedżerze kolejek. Aby możliwe było użycie wystarczającej liczby znaków, gwiazdka jest poprawna tylko na pozycjach od 1 do 33. Po gwiazdce nie mogą występować znaki inne niż spacje lub znak o kodzie zero.

Jest ona poprawna, aby gwiazdka wystąpiła na pierwszej pozycji znaku, w którym to przypadku nazwa składa się wyłącznie z znaków wygenerowanych przez menedżer kolejek.

W systemie z/OSnie należy używać nazwy z gwiazdką w pierwszej pozycji znaku, ponieważ w kolejce nie mogą być wykonywane sprawdzenia zabezpieczeń o pełnej nazwie, która jest generowana automatycznie.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest określana przez środowisko:

- W systemie z/OSwartością jest 'CSQ.*'.
- Na innych platformach wartością jest 'AMQ.*'.

Wartość jest łańcuchem zakończonym znakiem o kodzie zero w języku C, a pusty łańcuch dopełniony w innych językach programowania.

Liczba InvalidDest(MQLONG)

Jest to liczba kolejek z listy dystrybucyjnej, które nie zostały pomyślnie otwarte. Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Uwaga: Jeśli jest obecny, to pole jest ustawione *tylko*, jeśli parametr *CompCode* w wywołaniu MQOPEN lub MQPUT1 ma wartość MQCC_OK lub MQCC_WARNING; *nie* jest ustawiony, jeśli parametr *CompCode* ma wartość MQCC_FAILED.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_1.

Licznik KnownDest(MQLONG)

Jest to liczba kolejek znajdujących się na liście dystrybucyjnej, które są rozstrzygane do kolejek lokalnych i które zostały pomyślnie otwarte. Liczba ta nie obejmuje kolejek rozstrzyganych do kolejek zdalnych (nawet jeśli początkowo używana jest lokalna kolejka transmisji do przechowywania komunikatu). Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_1.

ObjectName (MQCHAR48)

Jest to nazwa lokalna obiektu zdefiniowana w menedżerze kolejek identyfikowanego przez produkt *ObjectQMgrName*. Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (od A do Z)
- Małe litery alfabetu (od a do z)
- Cyfry cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Użyj znaku o kodzie zero, aby wskazać koniec znaczących danych w nazwie; wartość NULL i wszystkie znaki po nim są traktowane jako znaki puste. W środowiskach wskazanych poniżej obowiązują następujące ograniczenia:

- W systemach, w których używane jest kodowanie EBCDIC Katakana, nie można używać małych liter.
- wł.z/OS:
 - Należy unikać nazw, które rozpoczynają się lub kończą znakiem podkreślenia. Nie mogą one być przetwarzane przez panele kontrolne i operacyjne.
 - Znak procentu ma specjalne znaczenie dla RACF. Jeśli produkt RACF jest używany jako zewnętrzny menedżer zabezpieczeń, nazwy nie mogą zawierać wartości procentowej. W takim przypadku nazwy te nie są uwzględniane podczas sprawdzania zabezpieczeń, gdy używane są profile ogólne produktu RACF.

- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w znaki cudzysłowu, gdy są określone w komendach. Tych znaków cudzysłowu nie należy określać dla nazw, które występują jako pola w strukturach lub jako parametry wywołań.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

Następujące punkty mają zastosowanie do typów wskazanych obiektów:

- Jeśli *ObjectName* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej, a zwraca w polu *ObjectName* nazwę utworzonej kolejki. Kolejka modelowa może być określona tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1.
- Jeśli *ObjectName* jest nazwą kolejki aliasowej z typem TARGTYPE (TOPIC), najpierw w nazwanej kolejce aliasowej wykonywane jest sprawdzenie zabezpieczeń. Jest to normalne, gdy używane są kolejki aliasowe. Gdy sprawdzanie zabezpieczeń zakończy się pomyślnie, wywołanie MQOPEN będzie kontynuowane i będzie zachowywać się jak wywołanie MQOPEN w tabeli MQOT_TOPIC;. Obejmuje to wykonanie sprawdzenia zabezpieczeń dla obiektu tematu administracyjnego.
- Jeśli *ObjectName* i *ObjectQMGrName* identyfikują kolejkę współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, nie może istnieć również definicja kolejki o tej samej nazwie w lokalnym menedżerze kolejek. Jeśli istnieje taka definicja (kolejka lokalna, kolejka aliasowa, kolejka zdalna lub kolejka modelowa), wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_NOT_UNIQUE.
- Jeśli otwierany obiekt jest listą dystrybucyjną (to znaczy *RecsPresent* jest obecna i większa od zera), *ObjectName* musi być pusty lub łańcuch pusty. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_NAME_ERROR.
- Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, mają zastosowanie reguły specjalne; w tym przypadku nazwa musi być całkowicie pusta w górę do pierwszego znaku o kodzie zero lub na końcu pola.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ObjectName* jest nazwą kolejki modelowej, a pole tylko wejściowe we wszystkich innych przypadkach. Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Nazwa *ObjectQMGr*(MQCHAR48)

Jest to nazwa menedżera kolejek, w którym zdefiniowany jest obiekt *ObjectName*. Znaki, które są poprawne w nazwie, są takie same jak w przypadku produktu *ObjectName* (patrz [“ObjectName \(MQCHAR48\)”](#) na stronie 483). Nazwa, która jest całkowicie pusta, do pierwszego znaku o wartości NULL lub do końca pola oznacza menedżer kolejek, z którym połączona jest aplikacja (lokalny menedżer kolejek).

Następujące punkty mają zastosowanie do typów wskazanych obiektów:

- Jeśli parametr *ObjectType* ma wartość MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS lub MQOT_Q_MGR, *ObjectQMGrName* musi być pusta lub musi być nazwą lokalnego menedżera kolejek.
- Jeśli *ObjectName* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej, a zwraca w polu *ObjectQMGrName* nazwę menedżera kolejek, w którym tworzona jest kolejka. Jest to nazwa lokalnego menedżera kolejek. Kolejka modelowa może być określona tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1.
- Jeśli *ObjectName* jest nazwą kolejki klastra, a *ObjectQMGrName* jest pusta, to miejsce docelowe komunikatów wysyłanych za pomocą uchwytu kolejki zwracanego przez wywołanie MQOPEN jest wybierane przez menedżer kolejek (lub wyjście obciążenia klastra, jeśli jest zainstalowane) w następujący sposób:

- Jeśli określono parametr `MQOO_BIND_ON_OPEN`, menedżer kolejek wybiera określoną instancję kolejki klastra podczas przetwarzania wywołania `MQOPEN`, a wszystkie komunikaty umieszczone przy użyciu tego uchwytu kolejki są wysyłane do tej instancji.
- Jeśli określono parametr `MQOO_BIND_NOT_FIXED`, menedżer kolejek może wybrać inną instancję kolejki docelowej (rezydując w innym menedżerze kolejek w klastrze) dla każdej kolejnej wywołania `MQPUT`, który używa tego uchwytu kolejki.

Jeśli aplikacja musi wysłać komunikat do *konkretnej* instancji kolejki klastra (czyli instancji kolejki znajdującej się w określonym menedżerze kolejek w klastrze), aplikacja musi określić nazwę tego menedżera kolejek w polu `ObjectQMGrName`. Wymusza wysłanie komunikatu przez lokalny menedżer kolejek do określonego docelowego menedżera kolejek.

- Jeśli `ObjectName` jest nazwą kolejki współużytkowanej, która jest własnością grupy współużytkowania kolejki, do której należy lokalny menedżer kolejek, `ObjectQMGrName` może być nazwą grupy współużytkowania kolejki, nazwą lokalnego menedżera kolejek lub pustą; komunikat jest umieszczany w tej samej kolejce, w zależności od tego, która z tych wartości jest określona.

Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS.

- Jeśli `ObjectName` jest nazwą kolejki współużytkowanej, której właścicielem jest zdalna grupa współużytkowania kolejek (to znaczy grupa współużytkowania kolejki, do której lokalny menedżer kolejek *nie* jest należy), `ObjectQMGrName` musi być nazwą grupy współużytkowania kolejki. Można użyć nazwy menedżera kolejek, który należy do tej grupy, ale może to opóźnić ten komunikat, jeśli dany menedżer kolejek nie jest dostępny po nadejściu komunikatu do grupy współużytkowania kolejki.
- Jeśli otwierany obiekt jest listą dystrybucyjną (to znaczy `RecsPresent` jest większa od zera), wartość `ObjectQMGrName` musi być pusta lub zawierać łańcuch pusty. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny `MQRC_OBJECT_Q_MGR_NAME_ERROR`.

Jest to pole wejściowe/wyjściowe dla wywołania `MQOPEN`, gdy `ObjectName` jest nazwą kolejki modelowej, a pole tylko wejściowe we wszystkich innych przypadkach. Długość tego pola jest podana przez wartość `MQ_Q_MGR_NAME_LENGTH`. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

ObjectRecPrzesunięcie (MQLONG)

Jest to przesunięcie w bajtach pierwszego rekordu obiektu `MQOR` od początku struktury `MQOD`.

Przesunięcie może być dodatnie lub ujemne. `ObjectRecOffset` jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość `RecsPresent` wynosi zero.

Gdy lista dystrybucyjna jest otwierana, w celu określenia nazw kolejek docelowych na liście dystrybucyjnej należy podać tablicę jednej lub większej liczby rekordów obiektu `MQOR`. Można to zrobić na jeden z dwóch sposobów:

- Za pomocą pola przesunięcia `ObjectRecOffset`.

W takim przypadku aplikacja musi zadeklarować własną strukturę zawierającą tabelę `MQOD`, po której następuje tablica rekordów `MQOR` (wraz z wymaganą wieloma elementami tablicy), a następnie ustaw `ObjectRecOffset` na przesunięcie pierwszego elementu w tablicy od początku tabeli `MQOD`. Upewnij się, że to przesunięcie jest poprawne i ma wartość, która może być zakwaterowana w tabeli `MQLONG` (najbardziej restrykcyjnym językiem programowania jest język COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

W przypadku języków programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który nie jest przenośny dla różnych środowisk (na przykład w języku programowania COBOL), należy użyć produktu `ObjectRecOffset`.

- Za pomocą pola wskaźnika `ObjectRecPtr`.

W takim przypadku aplikacja może zadeklarować tablicę struktur `MQOR` niezależnie od struktury `MQOD`, a następnie ustawić wartość `ObjectRecPtr` na adres tablicy.

W przypadku języków programowania, które obsługują typ danych wskaźnika w sposób przenośny do różnych środowisk (na przykład język programowania w języku C), należy użyć programu `ObjectRecPtr`.

Niezależnie od wybranej techniki, należy użyć jednej z następujących opcji: *ObjectRecOffset* i *ObjectRecPtr*; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_RECORDS_ERROR, jeśli oba są równe zero, albo oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_2.

ObjectRecPtr (MQPTR)

Jest to adres pierwszego rekordu obiektu MQOR. *ObjectRecPtr* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Aby określić rekordy obiektów, ale nie oba, można użyć opcji *ObjectRecPtr* lub *ObjectRecOffset*. Opis pola *ObjectRecOffset* znajduje się w sekcji [“ObjectRecPrzesunięcie \(MQLONG\)”](#) na stronie 485. Jeśli produkt *ObjectRecPtr* nie jest używany, ustaw go na pusty wskaźnik lub zerową liczbę bajtów.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_2.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtów o wartości all-null.

ObjectString (MQCHARV)

Pole *ObjectString* określa długą nazwę obiektu.

Określa długą nazwę obiektu, który ma być używany. To pole jest przywoływane tylko dla określonych wartości *ObjectType*; jest ignorowane w przypadku wszystkich innych wartości. Patrz opis *ObjectType*, aby uzyskać szczegółowe informacje o tym, które wartości wskazują, że to pole jest używane.

Jeśli wartość *ObjectString* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_STRING_ERROR.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQCHARV.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

ObjectType (MQLONG)

Typ obiektu nazwanego w deskrytorze obiektu. Dozwolone są następujące wartości:

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta. Nazwa obiektu znajduje się w polu *ObjectName*.

MQOT_Q

do kolejki błędów. Nazwa obiektu znajduje się w polu *ObjectName*.

MQOT_NAMELIST,

Lista nazw. Nazwa obiektu znajduje się w polu *ObjectName*.

MQOT_PROCESS

Definicja procesu. Nazwa obiektu znajduje się w polu *ObjectName*.

MQOT_Q_MGR

menedżerze kolejek. Nazwa obiektu znajduje się w polu *ObjectName*.

MQOT_TOPIC

. Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ObjectName* i *ObjectString*.

Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQOT_Q.

RecsPresent (MQLONG)

Jest to liczba rekordów obiektów MQOR, które zostały udostępnione przez aplikację. Jeśli ta liczba jest większa od zera, oznacza to, że lista dystrybucyjna jest otwierana, a *RecsPresent* jest liczbą kolejek docelowych na liście. Lista dystrybucyjna może zawierać tylko jedno miejsce docelowe.

Wartość parametru *RecsPresent* nie może być mniejsza niż zero, a jeśli jest większa niż zero *ObjectType* musi być równa MQOT_Q; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RECS_PRESENT_ERROR, jeśli te warunki nie są spełnione.

W systemie z/OS to pole musi być równe zero.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_2.

Łańcuch ResObject(MQCHARV)

Pole *ResObjectString* to długa nazwa obiektu po przetłumaczonej nazwie podanej w polu *ObjectName*.

To pole jest zwracane tylko w przypadku tematów i aliasów kolejek, które odwołują się do obiektu tematu.

Jeśli długa nazwa obiektu jest dostępna w produkcie *ObjectString*, a w produkcie *ObjectName* jest udostępniana żadna wartość, to wartość zwrócona w tym polu jest taka sama, jak podana w składce *ObjectString*.

Jeśli to pole zostanie pominięte (to znaczy *ResObjectString.VSBufSize* ma wartość zero), to pole *ResObjectString* nie zostanie zwrócone, ale długość zostanie zwrócona w *ResObjectString.VSLength*.

Jeśli długość buforu (podana w polu *ResObjectString.VSBufSize*) jest krótsza niż pełny *ResObjectString*, łańcuch zostanie obcięty i zwróci tyle znaków z prawej strony, co może zmieścić się w udostępnionym buforze.

Jeśli wartość *ResObjectString* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RES_OBJECT_STRING_ERROR.

Nazwa ResolvedQMgr(MQCHAR48)

Jest to nazwa docelowego menedżera kolejek po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *ResolvedQName*. *ResolvedQMgrName* może być nazwą lokalnego menedżera kolejek.

Jeśli *ResolvedQName* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, *ResolvedQMgrName* jest nazwą grupy współużytkowania kolejki. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, *ResolvedQName* może być nazwą grupy współużytkowania kolejki lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejki (rodzaj zwróconej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *ResolvedQMgrName* jest pusty:

- To nie jest kolejka
- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.
- Kolejka klastrowa z podaną wartością MQOO_BIND_NOT_FIXED (lub z wartością MQOO_BIND_AS_Q_DEF w momencie, gdy atrybut kolejki *DefBind* ma wartość MQBND_BIND_NOT_FIXED).
- Lista dystrybucyjna

To jest pole wyjściowe. Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_3.

ResolvedQName (MQCHAR48)

Jest to nazwa kolejki docelowej po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa kolejki, która istnieje w menedżerze kolejek identyfikowanego przez produkt *ResolvedQMgrName*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *ResolvedQName* jest pusty:

- To nie jest kolejka
- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.
- Lista dystrybucyjna
- Kolejka aliasowa, która odwołuje się do obiektu tematu (zamiast niej należy odwołać się do obiektu *ResObjectString*).
- Kolejka aliasowa, która jest tłumaczona na obiekt tematu.

To jest pole wyjściowe. Długość tego pola jest podana przez wartość *MQ_Q_NAME_LENGTH*. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż *MQOD_VERSION_3*.

ResolvedType (MQLONG)

Typ otwartego (podstawowego) obiektu, który jest otwierany.

Możliwe wartości:

MQOT_Q

Rozstrzygnięty obiekt jest kolejką. Ta wartość ma zastosowanie, gdy kolejka jest otwierana bezpośrednio lub gdy kolejka aliasowa wskazująca kolejkę jest otwierana.

MQOT_TOPIC

Rozstrzygnięty obiekt jest tematem. Ta wartość ma zastosowanie, gdy temat jest otwierany bezpośrednio lub gdy otwarto kolejkę aliasową wskazującą na obiekt tematu.

MQOT_NONE

Rozstrzygnięty typ nie jest ani kolejką, ani tematem.

ResponseRecPrzesunięcie (MQLONG)

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi *MQRR* od początku struktury *MQOD*. Przesunięcie może być dodatnie lub ujemne. *ResponseRecOffset* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Po otwarciu listy dystrybucyjnej można podać tablicę co najmniej jednego rekordu odpowiedzi *MQRR*, aby zidentyfikować kolejki, których otwarcie nie powiodło się (pole *CompCode* w *MQRR*), oraz przyczynę każdego niepowodzenia (pole *Reason* w tabeli *MQRR*). Dane są zwracane w tablicy rekordów odpowiedzi w tej samej kolejności, w jakiej znajdują się nazwy kolejek w tablicy rekordów obiektów. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (oznacza to, że niektóre kolejki zostały otwarte pomyślnie, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn); kod przyczyny *MQRC_MULTIPLE_UZASADNIENIE* wywołania wskazuje tę sprawę. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna ta jest zwracana w parametrze *Reason* wywołania *MQOPEN* lub *MQPUT1*, a rekordy odpowiedzi nie są ustawione. Rekordy odpowiedzi są opcjonalne, ale jeśli są one podane, muszą być z nich *RecsPresent*.

Rekordy odpowiedzi mogą być udostępniane w taki sam sposób, jak rekordy obiektów, poprzez określenie przesunięcia w składce *ResponseRecOffset* lub przez podanie adresu w programie *ResponseRecPtr*; Szczegółowe informacje na temat sposobu wykonania tej czynności zawiera sekcja "ObjectRecPrzesunięcie (MQLONG)" na stronie 485. Jednak nie można użyć więcej niż jednego z produktów *ResponseRecOffset* i *ResponseRecPtr*; wywołanie kończy się niepowodzeniem z kodem przyczyny *MQRC_RESPONSE_RECORDS_ERROR*, jeśli oba są niezerowe.

W przypadku wywołania *MQPUT1* te rekordy odpowiedzi są używane do zwracania informacji o błędach, które występują, gdy komunikat jest wysyłany do kolejek na liście dystrybucyjnej, a także błędów,

które występują podczas otwierania kolejek. Kod zakończenia i kod przyczyny z operacji put dla kolejki zastępują te operacje z operacji otwarcia dla tej kolejki tylko wtedy, gdy kod zakończenia z tej ostatniej operacji to MQCC_OK lub MQCC_WARNING.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_2.

ResponseRecPtr (MQPTR)

Jest to adres pierwszego rekordu odpowiedzi MQRR. *ResponseRecPtr* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Użyj opcji *ResponseRecPtr* lub *ResponseRecOffset*, aby określić rekordy odpowiedzi, ale nie oba te rekordy; szczegółowe informacje na ten temat zawiera sekcja [“ResponseRecPrzesunięcie \(MQLONG\)”](#) na stronie 488. Jeśli produkt *ResponseRecPtr* nie jest używany, ustaw go na pusty wskaźnik lub zerową liczbę bajtów.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_2.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtów o wartości all-null.

SelectionString (MQCHARV)

Jest to łańcuch używany do udostępniania kryteriów wyboru używanych podczas pobierania komunikatów z kolejki.

Produkt *SelectionString* nie może być udostępniany w następujących przypadkach:

- Jeśli *ObjectType* nie jest typu MQOT_Q
- Jeśli otwierana kolejka nie jest otwierana za pomocą jednej z opcji MQOO_BROWSE, lub MQOO_INPUT_*

Jeśli w takich przypadkach zostanie podana wartość *SelectionString*, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_SELECTOR_INVALID_FOR_TYPE.

Jeśli wartość *SelectionString* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury [“MQCHARV-łańcuch o zmiennej długości”](#) na stronie 298 lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SELECTION_STRING_ERROR. Maksymalna długość parametru *SelectionString* to MQ_SELECTOR_LENGTH.

Użycie produktu *SelectionString* jest opisane w sekcji [Selektory](#).

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQOD_STRUC_ID

Identyfikator struktury deskryptora obiektu.

Dla języka programowania C zdefiniowana jest również stała MQOD_STRUC_ID_ARRAY; ma taką samą wartość jak MQOD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQOD_STRUC_ID.

Licznik UnknownDest(MQLONG)

Jest to liczba kolejek znajdujących się na liście dystrybucyjnej, które są rozstrzygane do kolejek zdalnych i które zostały pomyślnie otwarte. Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli parametr *Version* jest mniejszy niż MQOD_VERSION_1.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być jedną z następujących wartości:

MQOD_VERSION_1

Struktura deskryptora obiektu Version-1 .

MQOD_VERSION_2

Struktura deskryptora obiektu Version-2 .

MQOD_VERSION_3

Struktura deskryptora obiektu Version-3 .

MQOD_VERSION_4

Struktura deskryptora obiektu Version-4 .

Wszystkie wersje są obsługiwane we wszystkich środowiskach produktu IBM MQ V7.0 .

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól.

Następująca stała określa numer wersji bieżącej wersji:

MQOD_CURRENT_VERSION

Bieżąca wersja struktury deskryptora obiektu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQOD_VERSION_1.

Wartości początkowe i deklaracje języka dla MQOD

Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQOD_STRUC_ID	'0D¬¬'
<i>Version</i>	MQOD_VERSION_1	1
<i>ObjectType</i>	Kolejka MQOT_Q	1
<i>ObjectName</i>	Brak	Pusty łańcuch lub odstępy
<i>ObjectQMgrName</i>	Brak	Pusty łańcuch lub odstępy
<i>DynamicQName</i>	Brak	'CSQ.*' w systemie z/OS ; 'AMQ.*' inaczej
<i>AlternateUserId</i>	Brak	Pusty łańcuch lub odstępy
<i>RecsPresent</i>	Brak	0
<i>KnownDestCount</i>	Brak	0
<i>UnknownDestCount</i>	Brak	0
<i>InvalidDestCount</i>	Brak	0
<i>ObjectRecOffset</i>	Brak	0
<i>ResponseRecOffset</i>	Brak	0
<i>ObjectRecPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>ResponseRecPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>AlternateSecurityId</i>	MQSID_NONE	Wartości null
<i>ResolvedQName</i>	Brak	Pusty łańcuch lub odstępy
<i>ResolvedQMgrName</i>	Brak	Pusty łańcuch lub odstępy
<i>ObjectString</i>	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV

Nazwa pola	Nazwa stałej	Wartość stałej
<i>SelectionString</i>	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV
<i>ResObjectString</i>	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV
<i>ResolvedType</i>	MQOT_NONE	0
Uwagi: - Symbol – reprezentuje pojedynczy pusty znak. - Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. - W języku programowania C: zmienna makraParametr MQOD_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze: MQOD MyOD = {MQOD_DEFAULT};		

Deklaracja C

```
typedef struct tagMQOD MQOD;
struct tagMQOD {
    MQCHAR4      StrucId;           /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQLONG       ObjectType;       /* Object type */
    MQCHAR48     ObjectName;       /* Object name */
    MQCHAR48     ObjectQMgrName;   /* Object queue manager name */
    MQCHAR48     DynamicQName;    /* Dynamic queue name */
    MQCHAR12     AlternateUserId;  /* Alternate user identifier */
    /* Ver:1 */
    MQLONG       RecsPresent;      /* Number of object records present */
    MQLONG       KnownDestCount;   /* Number of local queues opened
                                   successfully */
    MQLONG       UnknownDestCount; /* Number of remote queues opened
                                   successfully */
    MQLONG       InvalidDestCount; /* Number of queues that failed to
                                   open */
    MQLONG       ObjectRecOffset;  /* Offset of first object record from
                                   start of MQOD */
    MQLONG       ResponseRecOffset; /* Offset of first response record
                                   from start of MQOD */
    MQPTR        ObjectRecPtr;     /* Address of first object record */
    MQPTR        ResponseRecPtr;   /* Address of first response record */
    /* Ver:2 */
    MQBYTE40     AlternateSecurityId; /* Alternate security identifier */
    MQCHAR48     ResolvedQName;     /* Resolved queue name */
    MQCHAR48     ResolvedQMgrName;  /* Resolved queue manager name */
    /* Ver:3 */
    MQCHARV      ObjectString;      /* Object Long name */
    MQCHARV      SelectionString;    /* Message Selector */
    MQCHARV      ResObjectString;    /* Resolved Long object name*/
    MQLONG       ResolvedType       /* Alias queue resolved
                                   object type */
    /* Ver:4 */
};
```

Deklaracja języka COBOL

```
** MQOD structure
10  MQOD.
** Structure identifier
15  MQOD-STRUCID                PIC X(4).
** Structure version number
15  MQOD-VERSION                PIC S9(9) BINARY.
** Object type
```

```

15 MQOD-OBJECTTYPE          PIC S9(9) BINARY.
** Object name
15 MQOD-OBJECTNAME          PIC X(48).
** Object queue manager name
15 MQOD-OBJECTQMGRNAME      PIC X(48).
** Dynamic queue name
15 MQOD-DYNAMICQNAME        PIC X(48).
** Alternate user identifier
15 MQOD-ALTERNATEUSERID     PIC X(12).
** Number of object records present
15 MQOD-RECSPRESENT         PIC S9(9) BINARY.
** Number of local queues opened successfully
15 MQOD-KNOWNDESTCOUNT     PIC S9(9) BINARY.
** Number of remote queues opened successfully
15 MQOD-UNKNOWNDESTCOUNT   PIC S9(9) BINARY.
** Number of queues that failed to open
15 MQOD-INVALIDDESTCOUNT   PIC S9(9) BINARY.
** Offset of first object record from start of MQOD
15 MQOD-OBJECTRECOFFSET     PIC S9(9) BINARY.
** Offset of first response record from start of MQOD
15 MQOD-RESPONSERECOFFSET   PIC S9(9) BINARY.
** Address of first object record
15 MQOD-OBJECTRECPtr        POINTER.
** Address of first response record
15 MQOD-RESPONSERECPtr      POINTER.
** Alternate security identifier
15 MQOD-ALTERNATESECURITYID PIC X(40).
** Resolved queue name
15 MQOD-RESOLVEDQNAME        PIC X(48).
** Resolved queue manager name
15 MQOD-RESOLVEDQMGRNAME     PIC X(48).
** Object Long name
15 MQOD-OBJECTSTRING.
** Address of variable length string
20 MQOD-OBJECTSTRING-VSPTR   POINTER.
** Offset of variable length string
20 MQOD-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Message Selector
15 MQOD-SELECTIONSTRING.
** Address of variable length string
20 MQOD-SELECTIONSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-SELECTIONSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-SELECTIONSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-SELECTIONSTRING-VSCCSID PIC S9(9) BINARY.
** Resolved Long object name
15 MQOD-RESOBJECTSTRING.
** Address of variable length string
20 MQOD-RESOBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQOD-RESOBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQOD-RESOBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQOD-RESOBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQOD-RESOBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Alias queue resolved object type
15 MQOD-RESOLVEDTYPE        PIC S9(9) BINARY.

```

Deklaracja PL/I

```

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 */

Deklaracja High Level Assembler

MQOD	DSECT	
MQOD_STRUCID	DS CL4	Structure identifier
MQOD_VERSION	DS F	Structure version number
MQOD_OBJECTTYPE	DS F	Object type
MQOD_OBJECTNAME	DS CL48	Object name
MQOD_OBJECTQMGRNAME	DS CL48	Object queue manager name
MQOD_DYNAMICQNAME	DS CL48	Dynamic queue name
MQOD_ALTERNATEUSERID	DS CL12	Alternate user identifier
MQOD_RECSPRESENT	DS F	Number of object records present
MQOD_KNOWNDESTCOUNT	DS F	Number of local queues opened successfully
* MQOD_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_OBJECTRECPtr	DS F	Address of first object record
MQOD_RESPONSERECPtr	DS F	Address of first response record
MQOD_ALTERNATESECURITYID	DS XL40	Alternate security identifier
MQOD_RESOLVEDQNAME	DS CL48	Resolved queue name
MQOD_RESOLVEDQMGRNAME	DS CL48	Resolved queue manager name
MQOD_OBJECTSTRING	DS F	Object Long name
MQOD_OBJECTSTRING_VSPTR	DS F	Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET	DS F	Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSIZE	DS F	size of buffer
MQOD_OBJECTSTRING_VSLENGTH	DS F	Length of variable length string
MQOD_OBJECTSTRING_VSCCSID	DS F	CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH	EQU *- MQOD_OBJECTSTRING	
	ORG MQOD_OBJECTSTRING	
MQOD_OBJECTSTRING_AREA	DS CL(MQOD_OBJECTSTRING_LENGTH)	
* MQOD_SELECTIONSTRING	DS F	Message Selector
MQOD_SELECTIONSTRING_VSPTR	DS F	Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET	DS F	Offset of variable length string

```

MQOD_SELECTIONSTRING_VSBUFFSIZE 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_VSBUFFSIZE 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
                                ORG MQOD
MQOD_AREA DS CL(MQOD_LENGTH)

```

Wizualna deklaracja podstawowa

```

Type MQOD
  StructId      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-rekord obiektu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 61. Pola w tabeli MQOR		
Pole	Opis	Temat
<i>ObjectName</i>	Nazwa obiektu	ObjectName
<i>ObjectQMgrName</i>	Nazwa menedżera kolejek obiektów	NazwaObjectQMgr

Przegląd produktu MQOR

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Użyj struktury MQOR, aby określić nazwę kolejki i nazwę menedżera kolejek dla pojedynczej kolejki docelowej. MQOR jest strukturą wejściową dla wywołań MQOPEN i MQPUT1.

Zestaw znaków i kodowanie: Dane w tabeli MQOR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek

podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniu MQOPEN, można otworzyć listę kolejek; ta lista jest nazywana *listą dystrybucyjną*. Każdy komunikat umieszczany przy użyciu uchwytu kolejki zwróconego przez wywołanie MQOPEN jest umieszczany na każdej z kolejek na liście, pod warunkiem, że kolejka została pomyślnie otwarta.

Pola dla tabeli MQOR

Struktura MQOR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ObjectName (MQCHAR48)

Jest to taka sama sytuacja, jak w przypadku pola *ObjectName* w strukturze MQOD (szczegółowe informacje na ten temat zawiera tabela MQOD), z tym wyjątkiem, że:

- Musi to być nazwa kolejki.
- Nie może to być nazwa kolejki modelowej.

To jest zawsze pole wejściowe. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Nazwa ObjectQMGr(MQCHAR48)

Jest to taka sama sytuacja, jak w przypadku pola *ObjectQMGrName* w strukturze MQOD (szczegółowe informacje na ten temat zawiera tabela MQOD).

To jest zawsze pole wejściowe. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Wartości początkowe i deklaracje języków dla tabeli MQOR

Tabela 62. Początkowe wartości pól w MQOR dla MQOR		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>ObjectName</i>	Brak	Pusty łańcuch lub odstępy
<i>ObjectQMGrName</i>	Brak	Pusty łańcuch lub odstępy
Uwagi: 1. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania. 2. W języku programowania C: zmienna makraParametr MQOR_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:<pre>MQOR MyOR = {MQOR_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQOR MQOR;  
struct tagMQOR {  
    MQCHAR48 ObjectName;        /* Object name */  
    MQCHAR48 ObjectQMGrName;    /* Object queue manager name */  
};
```

Deklaracja języka COBOL

```
**    MQOR structure
```

```

10 MQOR.
**   Object name
15 MQOR-OBJECTNAME      PIC X(48).
**   Object queue manager name
15 MQOR-OBJECTQMGRNAME PIC X(48).

```

Deklaracja PL/I

```

dcl
1 MQOR based,
3 ObjectName      char(48), /* Object name */
3 ObjectQMGrName char(48); /* Object queue manager name */

```

Wizualna deklaracja podstawowa

```

Type MQOR
  ObjectName      As String*48 'Object name'
  ObjectQMGrName As String*48 'Object queue manager name'
End Type

```

MQPD-deskryptor właściwości

W poniższej tabeli podsumowano pola w strukturze.

Tabela 63. Pola w tabeli MQPD		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje
<i>Support</i>	Wymagana obsługa właściwości komunikatu	Obsługa
<i>Context</i>	Kontekst komunikatu, do którego należy właściwość	Kontekst
<i>CopyOptions</i>	Opcje kopiowania, do których należy właściwość	CopyOptions

Przegląd produktu MQPD

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS i IBM MQ MQI clients.

Przeznaczenie: MQPD służy do definiowania atrybutów właściwości. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSETMP i parametrem wyjściowym w wywołaniu MQINQMP.

Zestaw znaków i kodowanie: Dane w programie MQPD muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola dla tabeli MQPD

Struktura MQPD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Kontekst (MQLONG)

W tym temacie opisano kontekst komunikatu, do którego należy właściwość.

Po odebraniu przez menedżer kolejek komunikatu zawierającego właściwość zdefiniowaną przez produkt IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną, menedżer kolejek koryguje wartość pola *Context*.

Można określić następujące opcje:

MQPD_USER_CONTEXT

Właściwość jest powiązana z kontekstem użytkownika.

Do ustawienia właściwości powiązanej z kontekstem użytkownika przy użyciu wywołania MQSETMP nie jest wymagana żadna specjalna autoryzacja.

W przypadku menedżera kolejek produktu IBM WebSphere MQ 7.0 właściwość powiązana z kontekstem użytkownika jest zapisywana w sposób opisany dla parametru MQOO_SAVE_ALL_CONTEXT. Wywołanie MQPUT z podaną wartością MQPMO_PASS_ALL_CONTEXT powoduje, że właściwość zostanie skopiowana z zapisanego kontekstu do nowego komunikatu.

Jeśli wcześniej opisana opcja nie jest wymagana, można użyć następującej opcji:

MQPD_NO_CONTEXT

Ta właściwość nie jest powiązana z kontekstem komunikatu.

Nierozpoznana wartość jest odrzucana przy użyciu kodu *Reason* MQRC_PD_ERROR.

Jest to pole wejściowe/wyjściowe w wywołaniu MQSETMP i w polu wyjściowym z wywołania MQINQMP. Początkowa wartość tego pola to MQPD_NO_CONTEXT.

CopyOptions (MQLONG)

W tym temacie opisano typy komunikatów, do których należy skopiować właściwość. To jest pole tylko dla danych wyjściowych dla rozpoznanych właściwości zdefiniowanych IBM MQ ; IBM MQ ustawia odpowiednią wartość.

Gdy menedżer kolejek odbierze komunikat zawierający zdefiniowaną właściwość IBM MQ , którą menedżer kolejek rozpoznaje jako niepoprawną, menedżer kolejek koryguje wartość pola *CopyOptions* .

Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

MQCOPY_FORWARD

Ta właściwość jest kopiowana do przekazywanego komunikatu.

MQCOPY_PUBLISH

Ta właściwość jest kopiowana do komunikatu odebranego przez subskrybenta, gdy jest publikowany komunikat.

ODPOWIEDŹ MQCOPY_REPLY

Ta właściwość jest kopiowana do komunikatu odpowiedzi.

RAPORT MQCOPY_REPORT

Ta właściwość jest kopiowana do komunikatu raportu.

MQCOPY_ALL

Ta właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

Opcja domyślna: W celu podania domyślnego zestawu opcji kopiowania można podać następującą opcję:

MQCOPY_DEFAULT

Ta właściwość jest kopiowana do wiadomości przesyłanej, do komunikatu raportu lub do komunikatu odebranego przez subskrybenta w momencie publikowania komunikatu.

Jest to równoznaczne z określeniem kombinacji opcji MQCOPY_FORWARD, a także MQCOPY_REPORT i MQCOPY_PUBLISH.

Jeśli żadna z opcji, które zostały opisane wcześniej, nie jest wymagana, użyj następującej opcji:

MQCOPY_BRAK

Tej wartości należy użyć, aby wskazać, że nie są określone żadne inne opcje kopiowania; programowo nie istnieje relacja między tą właściwością a kolejnymi komunikatami. Ta właściwość jest zawsze zwracana dla właściwości deskryptora komunikatu.

Jest to pole wejściowe/wyjściowe w wywołaniu MQSETMP i w polu wyjściowym z wywołania MQINQMP. Wartością początkową tego pola jest MQCOPY_DEFAULT.

Opcje (MQLONG)

Wartość musi być następująca:

MQPD_NONE

Nie określono opcji

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQPD_NONE.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

Identyfikator MQPD_STRUC_ID

Identyfikator struktury deskryptora właściwości.

Dla języka programowania w języku C jest również zdefiniowana stała **MQPD_STRUC_ID_ARRAY**. Ma ona taką samą wartość jak **MQPD_STRUC_ID**, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQPD_STRUC_ID**.

Obsługa (MQLONG)

W tym polu opisano poziom obsługi właściwości komunikatu wymagany przez menedżer kolejek, aby komunikat zawierający tę właściwość mógł zostać umieszczony w kolejce. Dotyczy to tylko właściwości zdefiniowanych w produkcie IBM MQ; obsługa wszystkich pozostałych właściwości jest opcjonalna.

Pole jest automatycznie ustawiane na poprawną wartość, jeśli właściwość zdefiniowana przez produkt IBM MQ jest znana przez menedżer kolejek. Jeśli ta właściwość nie zostanie rozpoznana, przypisywany jest parametr MQPD_SUPPORT_OPTIONAL. Po odebraniu przez menedżer kolejek komunikatu zawierającego właściwość zdefiniowaną przez produkt IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną, menedżer kolejek koryguje wartość pola *Support*.

Podczas ustawiania właściwości zdefiniowanej w produkcie IBM MQ za pomocą wywołania MQSETMP na uchwycie komunikatu, w którym została ustawiona opcja MQCMHO_NO_VALIDATION, produkt *Support* staje się polem wejściowym. Umożliwia to aplikacji umieszczanie właściwości zdefiniowanej w produkcie IBM MQ z poprawną wartością, jeśli właściwość nie jest obsługiwana przez połączonego menedżera kolejek, ale w przypadku, gdy komunikat ma być przetworzony w innym menedżerze kolejek.

Wartość MQPD_SUPPORT_OPTIONAL jest zawsze przypisywany do właściwości, które nie są właściwościami zdefiniowanymi w produkcie IBM MQ.

Jeśli menedżer kolejek produktu IBM WebSphere MQ 7.0, który obsługuje właściwości komunikatu, odbiera właściwość zawierającą nierozpoznaną wartość *Support*, właściwość ta jest traktowana tak, jakby:

- Wartość MQPD_SUPPORT_REQUIRED została określona, jeśli dowolna z nierozpoznanych wartości znajduje się w masce MQPD_REJECT_UNSUP_MASK.
- Wartość MQPD_SUPPORT_REQUIRED_IF_LOCAL została określona, jeśli dowolna z nierozpoznanych wartości znajduje się w masce MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.
- Opcja MQPD_SUPPORT_OPTIONAL została określona w inny sposób.

Jedna z następujących wartości jest zwracana przez wywołanie MQINQMP lub jedna z wartości, która może zostać określona podczas używania wywołania MQSETMP dla uchwytu komunikatu, w którym ustawiona jest opcja MQCMHO_NO_VALIDATION:

MQPD_SUPPORT_OPTIONAL

Ta właściwość jest akceptowana przez menedżera kolejek nawet wtedy, gdy nie jest obsługiwana.

Tę właściwość można usunąć, aby komunikat mógł przepływać do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywany do właściwości, które nie są zdefiniowane w produkcie IBM MQ.

MQPD_SUPPORT_REQUIRED

Wymagana jest obsługa właściwości. Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje zdefiniowanej przez produkt IBM MQ właściwości. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_UNSUPPORTED_PROPERTY.

MQPD_SUPPORT_REQUIRED_IF_LOCAL

Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje zdefiniowanej przez IBM MQ właściwości, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_UNSUPPORTED_PROPERTY.

Wywołanie MQPUT lub MQPUT1 powiedzie się, jeśli komunikat jest przeznaczony dla zdalnego menedżera kolejek.

To jest pole wyjściowe w wywołaniu MQINQMP i pole wejściowe w wywołaniu MQSETMP, jeśli uchwyt komunikatu został utworzony za pomocą zestawu opcji MQCMHO_NO_VALIDATION. Wartością początkową tego pola jest MQPD_SUPPORT_OPTIONAL.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQPD_VERSION_1

Struktura deskryptora właściwości Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQPD_CURRENT_VERSION

Bieżąca wersja struktury deskryptora właściwości.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQPD_VERSION_1**.

Wartości początkowe i deklaracje języków dla tabeli MQPD

Tabela 64. Początkowe wartości pól w tabeli MQPD		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	Identyfikator 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
Uwagi:		
1. W języku programowania C zmienna makra MQPD_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQPD MyPD = {MQPD_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
    MQCHAR4 StrucId;      /* Structure identifier */
    MQLONG  Version;      /* Structure version number */
    MQLONG  Options;      /* Options that control the action of
                           MQSETMP and MQINQMP */
    MQLONG  Support;      /* Property support option */
    MQLONG  Context;      /* Property context */
    MQLONG  CopyOptions; /* Property copy options */
};
```

Deklaracja języka COBOL

```
** MQPD structure
10 MQPD.
** Structure identifier
15 MQPD-STRUCID PIC X(4).
** Structure version number
15 MQPD-VERSION PIC S9(9) BINARY.
** Options that control the action of MQSETMP and
** MQINQMP
15 MQPD-OPTIONS PIC S9(9) BINARY.
** Property support option
15 MQPD-SUPPORT PIC S9(9) BINARY.
** Property context
15 MQPD-CONTEXT PIC S9(9) BINARY.
** Property copy options
15 MQPD-COPYOPTIONS PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQPD based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 Options      fixed bin(31),    /* Options that control the action
                                of MQSETMP and MQINQMP */
3 Support      fixed bin(31),    /* Property support option */
3 Context      fixed bin(31),    /* Property context */
3 CopyOptions  fixed bin(31);    /* Property copy options */
```

Deklaracja High Level Assembler

```
MQPD          DSECT
MQPD_STRUCID   DS    CL4      Structure identifier
MQPD_VERSION   DS    F        Structure version number
MQPD_OPTIONS   DS    F        Options that control the
*               action of MQSETMP and MQINQMP
MQPD_SUPPORT   DS    F        Property support option
MQPD_CONTEXT   DS    F        Property context
MQPD_COPYOPTIONS DS    F      Property copy options
MQPD_LENGTH    EQU    *-MQPD
MQPD_AREA      DS    CL(MQPD_LENGTH)
```

MQPMO-Put-message, opcje

W poniższej tabeli podsumowano pola w strukturze.

Tabela 65. Struktura MQPMO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje sterujące działaniem MQPUT i MQPUT1	Opcje
<i>Timeout</i>	Zarezerwowane	Limit czasu
<i>Context</i>	Uchwyt obiektu kolejki wejściowej	Kontekst
<i>KnownDestCount</i>	Liczba komunikatów wystanych pomyślnie do kolejek lokalnych	KnownDestLiczba
<i>UnknownDestCount</i>	Liczba komunikatów wystanych pomyślnie do kolejek zdalnych	UnknownDestLiczba

Tabela 65. Struktura MQPMO (kontynuacja)

Pole	Opis	Temat
<i>InvalidDestCount</i>	Liczba komunikatów, których nie można było wysłać	InvalidDestLiczba
<i>ResolvedQName</i>	Rozstrzygnięta nazwa kolejki docelowej	ResolvedQName
<i>ResolvedQMGrName</i>	Rozstrzygnięta nazwa docelowego menedżera kolejek	ResolvedQMGrNazwa
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQPMO_VERSION_2.		
<i>RecsPresent</i>	Liczba istniejących rekordów komunikatów lub rekordów odpowiedzi	RecsPresent
<i>PutMsgRecFields</i>	Flagi wskazujące, które pola MQPMR są obecne	PutMsgRecFields
<i>PutMsgRecOffset</i>	Przesunięcie pierwszego rekordu umieszczenia komunikatu od początku wywołania MQPMO	PutMsgRecOffset
<i>ResponseRecOffset</i>	Przesunięcie pierwszego rekordu odpowiedzi od początku wywołania MQPMO	PrzesunięcieResponseRec
<i>PutMsgRecPtr</i>	Adres pierwszego rekordu umieszczenia komunikatu	PutMsgRecPtr
<i>ResponseRecPtr</i>	Adres pierwszego rekordu odpowiedzi	ResponseRecPtr
Uwaga: Pozostałe pola są ignorowane, jeśli wartość <i>Version</i> jest mniejsza niż MQPMO_VERSION_3.		
<i>OriginalMsgHandle</i>	Pierwotny uchwyt komunikatu	Uchwyt OriginalMsg
<i>NewMsgHandle</i>	Nowy uchwyt komunikatu	UchwytNewMsg
<i>Action</i>	Typ wykonywanej operacji umieszczania oraz relacja między oryginalnym komunikatem określonym w polu <i>OriginalMsgHandle</i> a nowym komunikatem określonym w polu <i>NewMsgHandle</i> .	Działanie
<i>PubLevel</i>	Poziom subskrypcji ukierunkowany na publikację	PubLevel

Przegląd produktu MQPMO

Dostępność: Wszystkie systemy IBM MQ oraz klienci IBM MQ połączone z tymi systemami.

Cel: Struktura MQPMO umożliwia aplikacji określanie opcji, które sterują sposobem umieszczania komunikatów w kolejkach lub publikowanymi w tematach. Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQPUT i MQPUT1.

Wersja: Bieżąca wersja programu MQPMO to MQPMO_VERSION_3. Niektóre pola są dostępne tylko w niektórych wersjach produktu MQPMO. Jeśli zachodzi potrzeba obsługi portów aplikacji między kilkoma środowiskami, należy upewnić się, że wersja programu MQPMO jest spójna we wszystkich środowiskach. Pola, które istnieją tylko w określonych wersjach struktury, są identyfikowane jako takie w [“MQPMO-Put-message, opcje”](#) na stronie 500 i w opisach pól.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję programu MQPMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQPMO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *Version* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie: Dane w programie MQPMO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQPMO

Struktura MQPMO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Działanie (MQLONG)

Określa typ wykonywanej operacji umieszczania oraz relację między oryginalnym komunikatem określonym w polu Uchwyt OriginalMsga nowym komunikatem określonym przez pole Uchwyt NewMsg. Właściwości komunikatu są wybierane przez menedżer kolejek w zależności od wartości określonego działania.

Zawartość deskryptora komunikatu można podać przy użyciu parametru MsgDesc w wywołaniach MQPUT lub MQPUT1 . Alternatywnie można nie podawać parametru MsgDesc lub określić, że jest on wyjściowy-tylko poprzez włączenie MQPMO_MD_FOR_OUTPUT_ONLY w polu Options struktury MQPMO.

Jeśli parametr MsgDesc nie zostanie podany lub jeśli został określony jako tylko wyjściowy, to deskryptor komunikatu dla nowego komunikatu zostanie zapełniony z pól uchwytu komunikatu MQPMO, zgodnie z regułami opisanymi w tym temacie.

Ustawienie kontekstu i przekazywanie działań opisane w sekcji [Kontrolowanie informacji kontekstowych](#) są aktywne po skomponowaniu deskryptora komunikatu.

Jeśli zostanie podana niepoprawna wartość działania, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_ACTION_ERROR.

Można określić jedno z następujących działań:

MQACTP_NEW

Trwa umieszczanie nowego komunikatu, a żaden związek z poprzednim komunikatem nie jest określony przez program. Deskryptor komunikatu składa się z następujących elementów:

- Jeśli w wywołaniu MQPUT lub MQPUT1 zostanie podana wartość MsgDesc , a MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w tabeli MQPMO.Options, jest ona używana jako deskryptor komunikatu, który nie został zmodyfikowany.
- Jeśli wartość MsgDesc nie jest podana, lub MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w MQPMO.Options , a następnie menedżer kolejek generuje deskryptor komunikatu przy użyciu kombinacji właściwości z uchwytu OriginalMsgi uchwytu NewMsg. Wszystkie pola deskryptora komunikatu jawnie ustawione w nowym uchwycie komunikatu mają pierwszeństwo przed tymi w oryginalnym uchwycie komunikatu.

Dane komunikatu są pobierane z parametru MQPUT lub MQPUT1 Buffer.

MQACTP_FORWARD

Przesyłany jest wcześniej pobrany komunikat. Oryginalny uchwyt komunikatu określa komunikat, który został wcześniej pobrany.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolne w deskrytorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli w wywołaniu MQPUT lub MQPUT1 zostanie podana wartość MsgDesc , a MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w tabeli MQPMO.Options, jest ona używana jako deskryptor komunikatu, który nie został zmodyfikowany.
- Jeśli wartość MsgDesc nie jest podana, lub MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w MQPMO.Options , a następnie menedżer kolejek generuje deskryptor komunikatu przy użyciu kombinacji właściwości z uchwytu OriginalMsgi uchwytu NewMsg. Wszystkie pola deskryptora komunikatu jawnie ustawione w nowym uchwycie komunikatu mają pierwszeństwo przed tymi w oryginalnym uchwycie komunikatu.

- Jeśli w tabeli MQPMO.Options, a następnie są honorowane.

Właściwości komunikatu składają się w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają wartość MQCOPY_FORWARD w tabeli MQPD.CopyOptions
- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, który ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek szczególny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale wartość właściwości ma wartość NULL. W tym przypadku właściwość jest usuwana z komunikatu.

Dane komunikatu, które mają być przekazywane, są pobierane z parametru MQPUT lub MQPUT1 Buffer.

ODPOWIEDŹ MQACTP_REPLY

Odpowiedź jest tworzona na wcześniej pobranym komunikacie. Oryginalny uchwyt komunikatu określa komunikat, który został wcześniej pobrany.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolne w deskrypcorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli w wywołaniu MQPUT lub MQPUT1 zostanie podana wartość MsgDesc , a MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w tabeli MQPMO.Options, jest ona używana jako deskryptor komunikatu, który nie został zmodyfikowany.
- Jeśli wartość MsgDesc nie jest podana, lub MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w MQPMO.Options , a następnie pola początkowego deskryptora komunikatu są wybierane w następujący sposób:

Tabela 66. Transformacja uchwytu komunikatu odpowiedzi	
Pole w strukturze MQMD	Użyta wartość
Raport	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI i MQRO_DISCARD_MSG są ustawione: MQRO_DISCARD_MSG w przeciwnym razie MQRO_NONE
MsgType	MQMT_REPLY
Utrata ważności	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI jest ustawione: Skopiowano z komunikatu wejściowego w przeciwnym razie MQEI_UNLIMITED
Opinie	MQFB_NONE
MsgId	Jeśli ustawiono wartość MQPMO_NEW_MSG_ID, wykonaj następujące czynności: Generowany jest nowy identyfikator komunikatu else if MQRO_PASS_MSG_ID jest ustawiona: Skopiowano z komunikatu wejściowego w przeciwnym razie MQMI_NONE

Tabela 66. Transformacja uchwytu komunikatu odpowiedzi (kontynuacja)	
Pole w strukturze MQMD	Użyta wartość
CorrelId	Jeśli ustawiona jest wartość MQPMO_NEW_CORREL_ID: Generowany jest nowy identyfikator korelacji else if MQRO_COPY_MSG_ID_TO_CORREL_ID jest ustawiona: Skopiowano z pola MsgId w Komunikat wejściowy W przeciwnym razie, jeśli ustawiono wartość MQRO_PASS_CORREL_ID: Skopiowano z pola CorrelId w Komunikat wejściowy w przeciwnym razie MQCI_NONE
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Puste
GroupId	MQGI_NONE
Numer_kolejny_komunikatu	1
Depozycja	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_NIEZDEFINIOWANY

- Deskryptor komunikatu jest następnie modyfikowany przez nowy uchwyt komunikatu-wszystkie pola deskryptora komunikatu jawnie ustawione jako właściwości w nowym uchwycie komunikatu mają pierwszeństwo przed polami deskryptora komunikatu zgodnie z opisem poprzednio.

Właściwości komunikatu składają się w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają wartość MQCOPY_REPLY w tabeli MQPD.CopyOptions
- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, który ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek szczególny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale wartość właściwości ma wartość NULL. W tym przypadku właściwość jest usuwana z komunikatu.

Dane komunikatu, które mają zostać przekazane, są pobierane z parametru MQPUT/MQPUT1 Buffer.

RAPORT MQACTP_REPORT

Raport jest generowany w wyniku wcześniej pobranego komunikatu. Oryginalny uchwyt komunikatu określa komunikat, który powoduje wygenerowanie raportu.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolne w deskryptorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli w wywołaniu MQPUT lub MQPUT1 zostanie podana wartość MsgDesc , a MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w tabeli MQPMO.Options, jest ona używana jako deskryptor komunikatu, który nie został zmodyfikowany.

- Jeśli wartość MsgDesc nie jest podana, lub MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w MQPMO.Options , a następnie pola początkowego deskryptora komunikatu są wybierane w następujący sposób:

Tabela 67. Transformacja uchwytu komunikatu raportu	
Pole w strukturze MQMD	Użyta wartość
Raport	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI i MQRO_DISCARD_MSG są ustawione: MQRO_DISCARD_MSG w przeciwnym razie MQRO_NONE
MsgType	Raport_menedżera_mQMT
Utrata ważności	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI jest ustawione: Skopiowano z komunikatu wejściowego w przeciwnym razie MQEI_UNLIMITED
MsgId	Jeśli ustawiono wartość MQPMO_NEW_MSG_ID, wykonaj następujące czynności: Generowany jest nowy identyfikator komunikatu else if MQRO_PASS_MSG_ID jest ustawiona: Skopiowano z komunikatu wejściowego w przeciwnym razie MQMI_NONE
CorrelId	Jeśli ustawiona jest wartość MQPMO_NEW_CORREL_ID: Generowany jest nowy identyfikator korelacji else if MQRO_COPY_MSG_ID_TO_CORREL_ID jest ustawiona: Skopiowano z pola MsgId w Komunikat wejściowy W przeciwnym razie, jeśli ustawiono wartość MQRO_PASS_CORREL_ID: Skopiowano z pola CorrelId w Komunikat wejściowy w przeciwnym razie MQCI_NONE
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Puste
OriginalLength	Ustaw na <i>BufferLength</i>

- Deskryptor komunikatu jest następnie modyfikowany przez nowy uchwyt komunikatu-wszystkie pola deskryptora komunikatu jawnie ustawione jako właściwości w nowym uchwycie komunikatu mają pierwszeństwo przed polami deskryptora komunikatu zgodnie z opisem poprzednio.

Właściwości komunikatu składają się w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają wartość MQCOPY_REPORT w tabeli MQPD.CopyOptions

- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, który ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek szczególny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale wartość właściwości ma wartość NULL. W tym przypadku właściwość jest usuwana z komunikatu.

Pole Feedback w wynikanej MQMD reprezentuje raport, który ma zostać wygenerowany. Wartość sprzężenia zwrotnego MQFB_NONE powoduje, że wywołanie MQPUT lub MQPUT1 nie powiodło się z kodem przyczyny MQRC_FEEDBACK_ERROR.

Aby wybrać dane użytkownika w komunikacie raportu, program IBM MQ konsultuje się z polami raportu i opinii w wynikanych MQMD, a parametry Bufor i BufferLength wywołania MQPUT lub MQPUT1.

- Jeśli wartością opcji Feedback jest MQFB_COA, MQFB_COD lub MQFB_EXPIRATION, to wartość raportu jest sprawdzana.
- Jeśli dowolny z poniższych przypadków ma wartość true, używane są pełne dane komunikatu z buforu o długości BufferLength.
 - Informacja zwrotna to MQFB_EXPIRATION i raport zawiera MQRO_EXPIRATION_WITH_FULL_DATA.
 - Informacja zwrotna to MQFB_COD i raport zawiera MQRO_COD_WITH_FULL_DATA.
 - Informacja zwrotna to MQFB_COA, a raport zawiera MQRO_COA_WITH_FULL_DATA
- Jeśli dowolny z poniższych przypadków ma wartość true, używane są pierwsze 100 bajtów komunikatu (lub BufferLength, jeśli jest to mniej niż 100) z buforu.
 - Informacja zwrotna to MQFB_EXPIRATION i raport zawiera MQRO_EXPIRATION_WITH_DATA.
 - Informacja zwrotna to MQFB_COD, a raport zawiera MQRO_COD_WITH_DATA.
 - Informacja zwrotna to MQFB_COA, a raport zawiera MQRO_COA_WITH_DATA.
- Jeśli funkcja Feedback ma wartość MQFB_EXPIRATION, MQFB_COD lub MQFB_COA, a raport nie zawiera opcji *_WITH_FULL_DATA lub *_WITH_DATA, które mają znaczenie dla tej wartości Feedback, dane użytkownika nie są dołączane do komunikatu.
- Jeśli dane zwrotne mają inną wartość niż wymienione powyżej, to bufor i BufferLength są używane jako normalne.

Wyprowadzenie danych użytkownika jest przedstawione w poniższej tabeli:

Kontekst (MQHOBJ)

Jeśli określono wartość MQPMO_PASS_IDENTITY_CONTEXT lub MQPMO_PASS_ALL_CONTEXT, to pole musi zawierać uchwyt kolejki wejściowej, z którego pobierane są informacje o kontekście, które mają być powiązane z umieszczonym komunikatem.

Jeśli nie określono wartości MQPMO_PASS_IDENTITY_CONTEXT ani MQPMO_PASS_ALL_CONTEXT, to pole jest ignorowane.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

Liczba InvalidDest(MQLONG)

Jest to liczba komunikatów, których nie można było wysłać do kolejek znajdujących się na liście dystrybucyjnej. Liczba ta obejmuje kolejki, których otwarcie nie powiodło się, a także kolejki, które zostały pomyślnie otwarte, ale dla których operacja put nie powiodła się. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Uwaga: To pole jest ustawiane, jeśli parametr *CompCode* w wywołaniu MQPUT lub MQPUT1 ma wartość MQCC_OK lub MQCC_WARNING; może zostać ustawiony, jeśli parametr *CompCode* ma wartość MQCC_FAILED, ale nie polegaj na tym w kodzie aplikacji.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_1.

To pole jest niezdefiniowane w produkcie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

Licznik KnownDest(MQLONG)

Jest to liczba komunikatów, które bieżące wywołanie MQPUT lub MQPUT1 zostało pomyślnie wysłane do kolejek na liście dystrybucyjnej, które są kolejkami lokalnymi. Liczba nie obejmuje komunikatów wysyłanych do kolejek, które są rozstrzygane do kolejek zdalnych (nawet jeśli początkowo używana jest lokalna kolejka transmisji do przechowywania komunikatu). To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_1.

To pole jest niezdefiniowane w produkcie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

Uchwyty NewMsg(MQHMSG)

Jest to opcjonalny uchwyt do umieszczanego komunikatu podlegający wartości w polu Działanie. Definiuje on właściwości komunikatu i zastępuje wartości *OriginalMsgHandle*, o ile są one określone.

W przypadku powrotu z wywołania funkcji **MQPUT** lub **MQPUT1** zawartość uchwytu odzwierciedla faktyczne działanie komunikatu.

To jest pole wejściowe. Wartością początkową tego pola jest **MQHM_NONE**. To pole jest ignorowane, jeśli wersja jest mniejsza niż **MQPMO_VERSION_3**.

Opcje MQPMO (MQLONG)

Pole Opcje steruje działaniem wywołań **MQPUT** i **MQPUT1** .

Opcja zasięgu. Istnieje możliwość określenia dowolnej lub żadnej z opcji MQPMO. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe). Należy zauważyć, że kombinacje, które nie są poprawne, są oznaczone; wszystkie pozostałe kombinacje są poprawne.

Następująca opcja steruje zakresem wysłanych publikacji:

MQPMO_SCOPE_QMGR

Publikacja jest wysyłana tylko do subskrybentów, którzy subskrybują ten menedżer kolejek. Publikacja nie jest przekazywana do żadnych zdalnych menedżerów kolejek publikowania/subskrypcji, które dokonały subskrypcji tego menedżera kolejek, co powoduje nadpisanie dowolnego zachowania ustawionego przy użyciu atrybutu tematu PUBSCOPE.

Uwaga: Jeśli ta opcja nie zostanie ustawiona, zasięg publikacji jest określany przez atrybut tematu PUBSCOPE.

Opcje publikowania. Następujące opcje sterują sposobem publikowania komunikatów w temacie:

MQPMO_SUPPRESS_REPLYTO

Wszystkie informacje określone w polach *ReplyToQ* i *ReplyToQMGR* deskryptora MQMD tej publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja jest używana z opcją raportu, która wymaga *ReplyToQ*, wywołanie nie powiedzie się i zostanie wykonane wywołanie MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETAIN

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Ten czas przechowywania pozwala subskrybentowi na żądanie kopii tej publikacji po jej opublikowaniu za pomocą wywołania MQSUBRQ. Umożliwia także wysyłanie publikacji do aplikacji, które dokonają subskrypcji po chwili publikacji tej publikacji (chyba że nie zostaną one wysłane za pomocą opcji

MQSO_NEW_PUBLICATIONS_ONLY). Jeśli aplikacja wysyła publikację, która została zachowana, jest ona wskazana przez właściwość komunikatu MQIsRetained w tej publikacji.

Tylko jedna publikacja może być przechowywana w każdym węźle drzewa tematów. Dlatego też, jeśli istnieje już zachowana publikacja dla tego tematu, opublikowana przez dowolną inną aplikację, zostanie ona zastąpiona tą publikacją. W związku z tym lepiej jest unikać posiadania więcej niż jednego publikatora zachowującego wiadomości na ten sam temat.

Jeśli subskrybent żąda zachowanych publikacji, użyta subskrypcja może zawierać znak wieloznaczny w temacie, w którym to przypadku może być zgodna liczba zachowanych publikacji (w różnych węzłach w drzewie tematów), a do aplikacji żądającej może być wysłanych kilka publikacji. Więcej informacji można znaleźć w opisie wywołania [“MQSUBRQ-żądanie subskrypcji”](#) na stronie 807 .

Więcej informacji na temat interakcji zachowanych publikacji z poziomami subskrypcji zawiera sekcja [Intercepting publications](#)(Przechwytywanie publikacji).

Jeśli ta opcja jest używana i nie można zachować publikacji, komunikat nie zostanie opublikowany, a wywołanie zakończy się niepowodzeniem z opcją MQRC_PUT_NOT_ZACHOWANE.

MQPMO_NOT_OWN_SUBS

Informuje menedżera kolejek o tym, że aplikacja nie chce wysyłać żadnych publikacji do subskrypcji, do których należy. Subskrypcje są uznawane za należące do tej samej aplikacji, jeśli uchwyt połączenia są takie same.

MQPMO_WARN_IF_NO_SUBS_MATCHED

Jeśli żadna subskrypcja nie jest zgodna z publikacją, zwróć kod zakończenia (*CompCode*) o wartości MQCC_WARNING i kod przyczyny MQRC_NO_SUBS_MATCHED.

Jeśli operacja put zwróci wartość MQRC_NO_SUBS_MATCHED, to publikacja nie została dostarczona do żadnej subskrypcji. Jeśli jednak w operacji put zostanie określona opcja MQPMO_RETAIN, komunikat zostanie zachowany i dostarczony do dowolnej późniejszej zdefiniowanej subskrypcji.

Subskrypcja tematu jest zgodna z publikacją, jeśli spełniony jest dowolny z następujących warunków:

- Komunikat jest dostarczany do kolejki subskrypcji.
- Komunikat został dostarczony do kolejki subskrypcji, ale problem z kolejką oznacza, że komunikat nie może zostać umieszczony w kolejce, a w konsekwencji został umieszczony w kolejce niedostarczanych komunikatów lub został usunięty.
- Zdefiniowano wyjście routingu, które pomija dostarczanie komunikatu do subskrypcji.

Subskrypcja tematu nie jest zgodna z publikacją, jeśli spełniony jest dowolny z następujących warunków:

- Subskrypcja zawiera łańcuch wyboru, który nie jest zgodny z publikacją.
- Subskrypcja określiła opcję MQSO_PUBLICATION_ON_REQUEST.
- Publikacja nie została dostarczona, ponieważ w operacji put została określona opcja MQPMO_NOT_OWN_SUBS, a subskrypcja jest zgodna z tożsamością publikatora.

Opcje punktu synchronizacji. Następujące opcje odnoszą się do udziału wywołania MQPUT lub MQPUT1 w ramach jednostki pracy:

MQPMO_SYNCPOINT

Żądanie ma działać w ramach normalnych protokołów jednostkowych pracy. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

Jeśli nie określono parametru MQPMO_SYNCPOINT i MQPMO_NO_SYNCPOINT, włączenie żądania umieszczania w protokołach jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek, a nie środowisko, w którym działa aplikacja. W systemie z/OS żądanie umieszczenia znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie umieszczenia nie znajduje się w obrębie jednostki pracy.

Z powodu tych różnic aplikacja, która ma być portem, nie może domyślnie zezwalać na tę opcję; należy jawnie określić wartość MQPMO_SYNCPOINT lub MQPMO_NO_SYNCPOINT.

Nie określaj MQPMO_SYNCPOINT z MQPMO_NO_SYNCPOINT.

MQPMO_NO_SYNCPOINT

Wniosek ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest dostępny natychmiast i nie można go usunąć, tworząc kopię zapasową jednostki pracy.

Jeśli nie określono wartości MQPMO_NO_SYNCPOINT i MQPMO_SYNCPOINT, włączenie żądania umieszczania w protokołach jednostki pracy jest określone przez środowisko, w którym działa menedżer kolejek, a nie środowisko, w którym działa aplikacja. W systemie z/OS żądanie umieszczenia znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie umieszczenia nie znajduje się w obrębie jednostki pracy.

Z powodu tych różnic aplikacja, która ma być portem, nie może domyślnie zezwalać na tę opcję; należy jawnie określić wartość MQPMO_SYNCPOINT lub MQPMO_NO_SYNCPOINT.

Nie określaj MQPMO_NO_SYNCPOINT z MQPMO_SYNCPOINT.

Opcje identyfikator-komunikatu i identyfikatora korelacji. Następujące opcje zwracają się do menedżera kolejek o wygenerowanie nowego identyfikatora komunikatu lub identyfikatora korelacji:

MQPMO_NEW_MSG_ID

Menedżer kolejek zastępuje zawartość pola *MsgId* w strukturze MQMD nowym identyfikatorem komunikatu. Ten identyfikator komunikatu jest wysyłany razem z komunikatem i jest zwracany do aplikacji na wyjściu z wywołania MQPUT lub MQPUT1.

Opcję MQPMO_NEW_MSG_ID można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej. Szczegółowe informacje można znaleźć w opisie pola *MsgId* w strukturze MQPMR.

Użycie tej opcji zwalnia z konieczności zresetowania pola *MsgId* na wartość MQMI_NONE przed każdym wywołaniem MQPUT lub MQPUT1.

MQPMO_NEW_CORREL_ID

Menedżer kolejek zastępuje treść pola *CorrelId* w strukturze MQMD z nowym identyfikatorem korelacji. Ten identyfikator korelacji jest wysyłany z komunikatem i zwracany do aplikacji na wyjściu z wywołania MQPUT lub MQPUT1.

Opcję MQPMO_NEW_CORREL_ID można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej; szczegóły informacji można znaleźć w opisie pola *CorrelId* w strukturze MQPMR.

MQPMO_NEW_CORREL_ID jest przydatne w sytuacjach, gdy aplikacja wymaga unikalnego identyfikatora korelacji.

Opcje grupy i segmentu. Poniższe opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Zapoznaj się z tymi definicjami, które pomogą Ci zrozumieć tę opcję.



Ostrzeżenie: Nie można używać segmentowanych lub zgrupowanych komunikatów z publikowania/subskrybowania.

Komunikat fizyczny

Jest to najmniejsza jednostka informacji, która może zostać umieszczona w kolejce lub usunięta z kolejki. Jest ona często zgodna z informacjami podanymi lub pobraną w pojedynczej operacji MQPUT, MQPUT1 lub MQGET. Każdy komunikat fizyczny ma własny deskryptor komunikatu (MQMD). Ogólnie, komunikaty fizyczne wyróżniają się różnymi wartościami dla identyfikatora komunikatu (pole *MsgId* w strukturze MQMD), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny

Komunikat logiczny jest pojedynczą jednostką informacji o aplikacji tylko dla platform innych niż z/OS. W przypadku braku ograniczeń systemowych komunikat logiczny jest taki sam, jak komunikat fizyczny. Jednak w przypadku, gdy komunikaty logiczne są bardzo duże, ograniczenia systemowe mogą być zalecane lub konieczne, aby podzielić komunikat logiczny na dwa lub więcej komunikatów fizycznych, zwanych *segmentami*.

Komunikat logiczny, który został posegmentowany, składa się z dwóch lub większej liczby komunikatów fizycznych o tym samym identyfikatorze grupy innej niż NULL (pole *GroupId*

w strukturze MQMD) i o tym samym numerze kolejnym komunikatu (pole *MsgSeqNumber* w strukturze MQMD). Segmenty są rozróżniane przez różne wartości dla przesunięcia segmentu (pole *Offset* w strukturze MQMD), co daje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dopuszczona przez aplikację wysyłającą, ma również identyfikator grupy o wartości innej niż NULL, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają identyfikator grupy o wartości NULL (MQGI_NONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy

Grupa komunikatów jest zestawem jednego lub większej liczby komunikatów logicznych, które mają taki sam identyfikator grupy, który nie ma wartości NULL. Komunikaty logiczne w grupie są rozróżniane przez różne wartości dla numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do *n*, gdzie *n* jest liczbą komunikatów logicznych w grupie. Jeśli co najmniej jeden komunikat logiczny jest segmentowany, w grupie jest więcej niż *n* komunikatów fizycznych.

MQPMO_LOGICAL_ORDER

Ta opcja informuje menedżera kolejek o tym, w jaki sposób aplikacja umieszcza komunikaty w grupach i segmentach komunikatów logicznych. Można ją określić tylko w wywołaniu MQPUT. Nie jest ona poprawna w wywołaniu metody MQPUT1.

Jeśli zostanie określona opcja MQPMO_LOGICAL_ORDER, oznacza to, że aplikacja używa kolejnych wywołań MQPUT w następujących celach:

1. Umieszczenie segmentów w każdym komunikacie logicznym w kolejności rosnącego przesunięcia segmentu, począwszy od 0, bez przerw.
2. Umieszczenie wszystkich segmentów w jednym komunikacie logicznym przed umieszczeniem segmentów w następnym komunikacie logicznym.
3. Umieszczenie komunikatów logicznych w każdej grupie komunikatów w kolejności rosnących numerów kolejnych komunikatów, począwszy od 1, bez przerw. Numer kolejny komunikatu jest zwiększany automatycznie w produkcie IBM MQ.
4. Umieszczenie wszystkich komunikatów logicznych w jednej grupie komunikatów przed umieszczeniem komunikatów logicznych w następnej grupie komunikatów.

Szczegółowe informacje na temat komendy MQPMO_LOGICAL_ORDER zawiera sekcja [uporządkowanie logiczne i fizyczne](#).

Opcje kontekstu. Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

MQPMO_NO_CONTEXT

Zarówno kontekst tożsamości, jak i kontekst źródłowy są ustawione w taki sposób, aby wskazywać brak. Oznacza to, że pola kontekstu w strukturze MQMD są ustawione na:

- Odstępy dla pól znakowych
- Wartości puste dla pól typu byte
- Zera dla pól liczbowych

MQPMO_DEFAULT_CONTEXT

Komunikat ma zawierać domyślne informacje o kontekście, które są z nim powiązane, zarówno dla tożsamości, jak i pochodzenia. Menedżer kolejek ustawia pola kontekstu w deskrypcorze komunikatu w następujący sposób:

Pole w strukturze MQMD

UserIdentifier

Użyta wartość

Określone w środowisku, jeśli jest to możliwe; w przeciwnym razie należy ustawić odstępy.

Pole w strukturze MQMD	Użyta wartość
<i>AccountingToken</i>	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie należy ustawić wartość MQACT_NONE.
<i>ApplIdentityData</i>	Ustaw wartość pustą.
<i>PutApplType</i>	Określane z poziomu środowiska.
<i>PutApplName</i>	Określone w środowisku, jeśli jest to możliwe; w przeciwnym razie należy ustawić odstępy.
<i>PutDate</i>	Ustaw datę, w której komunikat jest umieszczany.
<i>PutTime</i>	Służy do ustawiania czasu, w którym komunikat jest umieszczany.
<i>ApplOriginData</i>	Ustaw wartość pustą.

Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Jeśli nie określono opcji kontekstu, są to wartości domyślne i działania.

MQPMO_PASS_IDENTITY_CONTEXT

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Kontekst tożsamości jest przyjmowany z uchwytu kolejki określonego w polu *Context*. Informacje o kontekście pochodzenia są generowane przez menedżer kolejek w taki sam sposób, jak dla parametru MQPMO_DEFAULT_CONTEXT (patrz powyższa tabela dla wartości). Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

W przypadku wywołania MQPUT kolejka musi zostać otwarta z opcją MQOO_PASS_IDENTITY_CONTEXT (lub z opcją, która jej implikuje). Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją MQOO_PASS_IDENTITY_CONTEXT.

MQPMO_PASS_ALL_CONTEXT

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Kontekst jest przyjmowany z uchwytu kolejki określonego w polu *Context*. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontrolowanie informacji o kontekście](#).

Dla wywołania MQPUT kolejka musi zostać otwarta z opcją MQOO_PASS_ALL_CONTEXT (lub z opcją, która jej implikuje). Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją MQOO_PASS_ALL_CONTEXT.

MQPMO_SET_IDENTITY_CONTEXT

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Aplikacja określa kontekst tożsamości w strukturze MQMD. Informacje o kontekście pochodzenia są generowane przez menedżer kolejek w taki sam sposób, jak dla parametru MQPMO_DEFAULT_CONTEXT (patrz powyższa tabela dla wartości). Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

W przypadku wywołania MQPUT kolejka musi zostać otwarta z opcją MQOO_SET_IDENTITY_CONTEXT (lub z opcją, która jej implikuje). Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją MQOO_SET_IDENTITY_CONTEXT.

MQPMO_SET_ALL_CONTEXT

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Aplikacja określa tożsamość, pochodzenie i kontekst użytkownika w strukturze MQMD. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

W przypadku wywołania MQPUT kolejka musi zostać otwarta z opcją MQOO_SET_ALL_CONTEXT. Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją MQOO_SET_ALL_CONTEXT.

Można określić tylko jedną z opcji kontekstu MQPMO_*_CONTEXT. Jeśli nie zostanie podana żadna wartość, przyjmowana jest wartość MQPMO_DEFAULT_CONTEXT.

Opcje właściwości. Następująca opcja odnosi się do właściwości komunikatu:

MQPMO_MD_FOR_OUTPUT_ONLY

Parametr deskryptora komunikatu musi być używany tylko do wyjścia w celu zwrócenia deskryptora komunikatu, który został umieszczony w komunikacie. Pola deskryptora komunikatu powiązane z polami *NewMsgHandle*, *OriginalMsgHandle* lub obu pól struktury **MQPMO** muszą być używane do wprowadzania danych.

Jeśli nie zostanie podany poprawny uchwyt komunikatu, wywołanie zakończy się niepowodzeniem z kodem przyczyny **MQRC_MD_ERROR**.

Opcje umieszczania odpowiedzi. Następujące opcje kontrolują odpowiedź zwróconej do wywołania MQPUT lub MQPUT1. Można określić tylko jedną z tych opcji. Jeśli nie zostaną określone wartości MQPMO_ASYNC_RESPONSE i MQPMO_SYNC_RESPONSE, przyjmowana jest wartość MQPMO_RESPONSE_AS_Q_DEF lub MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_ASYNC_RESPONSE

Opcja MQPMO_ASYNC_RESPONSE żąda, aby operacja MQPUT lub MQPUT1 została zakończona bez oczekiwania aplikacji na zakończenie wywołania przez menedżer kolejek. Użycie tej opcji może zwiększyć wydajność przesyłania komunikatów, szczególnie w przypadku aplikacji korzystających z powiązań klienta. Aplikacja może okresowo sprawdzać, używając komendy MQSTAT, niezależnie od tego, czy wystąpił błąd podczas poprzednich wywołań asynchronicznych.

W przypadku tej opcji gwarantowane jest zakończenie tylko następujących pól w strukturze MQMD;

- Dane_tożsamości_aplikacji
- Typ_aplikacji_wstawiającej
- Nazwa_aplikacji_wstawiającej
- Dane_pochodzenia_aplikacji

Dodatkowo, jeśli jako opcje określono obie wartości: MQPMO_NEW_MSG_ID lub MQPMO_NEW_CORREL_ID, zwracane są również zwracane wartości MsgId i CorrelId. (MQPMO_NEW_MSG_ID może być określone niejawnie przez określenie pustego pola MsgId).

Zostaną zakończone tylko poprzednie określone pola. Inne informacje, które normalnie zostaną zwrócone w strukturze MQMD lub MQPMO, nie są zdefiniowane.

Podczas żądania asynchronicznej odpowiedzi put dla MQPUT1 wartości ResolvedQName i ResolvedQMgnazw zwracane w strukturze MQOD nie są zdefiniowane.

W przypadku żądania asynchronicznej odpowiedzi put dla operacji MQPUT lub MQPUT1, CompCode i przyczyna MQCC_OK i MQRC_NONE nie muszą oznaczać, że komunikat został pomyślnie umieszczony w kolejce. Podczas tworzenia aplikacji MQI, która korzysta z asynchronicznej odpowiedzi put, i wymaga potwierdzenia, że komunikaty zostały umieszczone w kolejce, należy sprawdzić zarówno kod CompCode, jak i kody przyczyny z operacji put, a także użyć komendy MQSTAT w celu wystąpienia zapytania o asynchroniczne informacje o błędach.

Mimo że powodzenie lub niepowodzenie poszczególnych wywołań MQPUT lub MQPUT1 nie są zwracane natychmiast, pierwszy błąd, który wystąpił w wywołaniu asynchronicznym, można określić później w wyniku wywołania MQSTAT.

Jeśli komunikat trwał w punkcie synchronizacji nie zostanie dostarczony przy użyciu asynchronicznej odpowiedzi put, a użytkownik podejmie próbę zatwierdzenia transakcji, zatwierdzenie nie powiedzie się, a transakcja zostanie wycofana z kodu zakończenia MQCC_FAILED i z powodu wywołania MQRC_BACKED_OUT. Aplikacja może wywołać wywołanie MQSTAT w celu określenia przyczyny niepowodzenia poprzedniej operacji MQPUT lub MQPUT1.

MQPMO_SYNC_RESPONSE

Określenie tego typu odpowiedzi powoduje, że operacja MQPUT lub MQPUT1 jest zawsze emitowana synchronicznie. Jeśli operacja put zakończy się pomyślnie, wszystkie pola w strukturze MQMD i MQPMO zostaną zakończone.

Ta opcja zapewnia odpowiedź synchroniczną bez względu na domyślną wartość odpowiedzi umieszczonej w obiekcie kolejki lub tematu.

MQPMO_RESPONSE_AS_Q_DEF

Jeśli wartość ta jest określona dla wywołania MQPUT, to użyty typ odpowiedzi jest przyjmowany z wartości DEFPRESP określonej w kolejce po pierwszym otwarciu przez aplikację. Jeśli aplikacja kliencka jest połączona z menedżerem kolejek na poziomie wcześniejszym niż Wersja 7.0, zachowuje się tak, jakby została określona wartość MQPMO_SYNC_RESPONSE.

Jeśli ta opcja jest określona dla wywołania MQPUT1, wartość atrybutu DEFPRESP nie jest znana, zanim żądanie zostanie wysłane do serwera. Domyślnie, jeśli wywołanie MQPUT1 używa obiektu MQPMO_SYNCPOINT, który zachowuje się jak w przypadku odpowiedzi MQPMO_ASYNC_RESPONSE, i jeśli używany jest parametr MQPMO_NO_SYNCPOINT, jest on zachowywał się tak, jak w przypadku MQPMO_SYNC_RESPONSE. Można jednak przestawić to zachowanie domyślne, ustawiając właściwość Put1DefaultAlwaysSync w pliku konfiguracyjnym klienta. Patrz sekcja [Seksja CHANNELS w pliku konfiguracyjnym klienta](#).

MQPMO_RESPONSE_AS_TOPIC_DEF

MQPMO_RESPONSE_AS_TOPIC_DEF to synonim komendy MQPMO_RESPONSE_AS_Q_QDEF do użycia z obiektami tematów.

Inne opcje. Następujące opcje sterują kontrolą autoryzacji, co się dzieje, gdy menedżer kolejek jest wygaszany, a także rozstrzyganie nazw kolejek i menedżerów kolejek:

MQPMO_ALTERNATE_USER_AUTHORITY

MQPMO_ALTERNATE_USER_AUTHORITY wskazuje, że pole *AlternateUserId* w parametrze *ObjDesc* wywołania MQPUT1 zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności uprawnień do umieszczania komunikatów w kolejce. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy produkt *AlternateUserId* jest uprawniony do otwarcia kolejki z określonymi opcjami, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma do tego uprawnienia. (Nie dotyczy to jednak określonych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym aplikacja jest uruchomiona).

Ta opcja jest poprawna tylko w przypadku wywołania MQPUT1.

MQPMO_FAIL_IF QUIESCING

Ta opcja wymusza niepowodzenie wywołania MQPUT lub MQPUT1, jeśli menedżer kolejek znajduje się w stanie wygaszania.

W systemie z/OsA opcja wymusza również, że wywołanie MQPUT lub MQPUT1 nie powiedzie się, jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wygaszania.

Wywołanie zwraca kod zakończenia MQCC_FAILED z kodem przyczyny MQRC_Q_MGR QUIESCING lub MQRC_CONNECTION QUIESCING.

MQPMO_RESOLVE_LOCAL_Q

Użyj tej opcji, aby wypełnić *ResolvedQName* w strukturze MQPMO nazwą kolejki lokalnej, do której jest umieszczany komunikat, oraz *ResolvedQMGrName* nazwą lokalnego menedżera kolejek, który udostępnia kolejkę lokalną. Więcej informacji na temat tabeli MQPMO_RESOLVE_LOCAL_Q zawiera temat [MQOO_RESOLVE_LOCAL_Q](#).

Jeśli użytkownik jest uprawniony do umieszczania w kolejce, ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQPUT. Nie jest wymagane żadne uprawnienie specjalne.

Opcja domyślna. Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

MQPMO_BRAK

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Parametr MQPMO_NONE jest zdefiniowany w dokumentacji programu pomocowego; nie jest przeznaczony, aby ta opcja była używana z innymi, ale jako że jej wartość jest równa zero, nie można wykryć takiego użycia.

Zmienna MQPMO_NONE jest polem wejściowym. Wartością początkową pola *Options* jest MQPMO_NONE.

Uchwyt komunikatu OriginalMsg(MQHMSG)

To jest opcjonalny uchwyt do komunikatu. Być może został on wcześniej pobrany z kolejki. Użycie tego uchwytu jest uzależnione od wartości pola *Action* (patrz także [Uchwyt NewMsg](#)).

Treść oryginalnego uchwytu komunikatu nie zostanie zmieniona za pomocą wywołania **MQPUT** lub **MQPUT1**.

To jest pole wejściowe. Wartością początkową tego pola jest **MQHM_NONE**. To pole jest ignorowane, jeśli wersja jest mniejsza niż **MQPMO_VERSION_3**.

PubLevel (MQLONG)

Wartością początkową tego pola jest 9. Poziom subskrypcji docelowej tej publikacji. Ta publikacja otrzymuje tylko te subskrypcje o najwyższym poziomie SubLevel mniejszym lub równym tej wartości. Wartość ta musi należeć do zakresu od zera do 9; zero oznacza najniższy poziom. Jeśli jednak publikacja została zachowana, nie jest ona już dostępna dla subskrybentów na wyższych poziomach, ponieważ jest ponownie publikowana na poziomie PubLevel 1.

Więcej informacji na ten temat zawiera sekcja [Intercepting publications](#).

PutMsgRecFields (MQLONG)

To pole zawiera flagi, które wskazują, które pola MQPMR są obecne w rekordach umieszczania komunikatów udostępnianych przez aplikację. Opcji *PutMsgRecFields* należy używać tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero, a zarówno *PutMsgRecOffset*, jak i *PutMsgRecPtr* są równe zero.

W przypadku pól, które są obecne, menedżer kolejek używa dla każdego miejsca docelowego wartości z pól w odpowiednim rekordzie komunikatu umieszczonego. W przypadku pól, które są nieobecne, menedżer kolejek używa wartości ze struktury MQMD.

Użyj co najmniej jednej z następujących opcji, aby wskazać, które pola są obecne w rekordach umieszczania komunikatów:

MQPMRF_MSG_ID,

Pole identyfikatora komunikatu jest obecne.

MQPMRF_CORREL_ID

Pole identyfikatora korelacji jest obecne.

Identyfikator MQPMRF_GROUP_ID

Pole identyfikatora grupy jest obecne.

MQPMRF_FEEDBACK

Pole informacji zwrotnej jest obecne.

MQPMRF_ACCOUNTING_TOKEN,

Pole tokenu rozliczania jest obecne.

Jeśli ta opcja zostanie podana, należy określić wartość **MQPMO_SET_IDENTITY_CONTEXT** lub **MQPMO_SET_ALL_CONTEXT** w polu *Options*. Jeśli ten warunek nie jest spełniony, wywołanie nie powiedzie się i zostanie podany kod przyczyny **MQRC_PMO_RECORD_FLAGS_ERROR**.

Jeśli nie ma żadnych pól MQPMR, można określić następujące elementy:

MQPMRF_NONE

Nie istnieją pola rekordu komunikatu umieszczonego w komunikacie.

Jeśli ta wartość jest określona, wartość *RecsPresent* musi być zerowa albo obie wartości *PutMsgRecOffset* i *PutMsgRecPtr* muszą mieć wartość zero.

Parametr **MQPMRF_NONE** jest zdefiniowany w celu uzyskania dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Jeśli program *PutMsgRecFields* zawiera flagi, które nie są poprawne, lub jeśli udostępnione są rekordy komunikatów, ale produkt *PutMsgRecFields* ma wartość **MQPMRF_NONE**, wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_PMO_RECORD_FLAGS_ERROR**.

To jest pole wejściowe. Wartością początkową tego pola jest MQPMRF_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG)

Jest to przesunięcie w bajtach pierwszego rekordu komunikatu umieszczonego w MQPMR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. *PutMsgRecOffset* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, w celu określenia określonych właściwości komunikatu dla każdego miejsca docelowego można podać tablicę jednego lub większej liczby rekordów komunikatów umieszczonych w tabeli MQPMR. Właściwości te są następujące:

- Identyfikator komunikatu
- Identyfikator korelacji
- Identyfikator grupy
- Wartość sprzężenia zwrotnego
- Token rozliczania

Nie ma potrzeby określania wszystkich tych właściwości, ale niezależnie od wybranego podzbioru, należy określić pola w poprawnej kolejności. Szczegółowe informacje można znaleźć w opisie struktury MQPMR.

Zwykle musi istnieć tyle rekordów umieszczania komunikatów, ponieważ istnieją rekordy obiektów określone przez MQOD, gdy lista dystrybucyjna jest otwarta; każdy rekord umieszczania komunikatów dostarcza właściwości komunikatu dla kolejki identyfikowanej przez odpowiedni rekord obiektu. Kolejki z listy dystrybucyjnej, które nie otwierają się, muszą nadal umieszczać dla nich rekordy komunikatów na odpowiednich pozycjach w tablicy, chociaż właściwości komunikatu są ignorowane w tym przypadku.

Liczba rekordów umieszczania komunikatów może być różna od liczby rekordów obiektów. Jeśli liczba rekordów umieszczania komunikatów jest mniejsza niż rekordy obiektów, to właściwości komunikatu dla miejsc docelowych, które nie zawierają rekordów komunikatów, są pobierane z odpowiednich pól w deskrytorze komunikatu MQMD. Jeśli rekordy komunikatów są umieszczane w większej ilości niż rekordy obiektów, nadmiarowe rekordy nie są używane (mimo że nadal musi istnieć możliwość ich uzyskania). Rekordy umieszczania komunikatów są opcjonalne, ale jeśli są one podane, muszą być z nich *RecsPresent*.

Udostępnij rekordy umieszczania komunikatów w podobny sposób do rekordów obiektów w MQOD, podając przesunięcie w *PutMsgRecOffset* lub podając adres w programie *PutMsgRecPtr*. aby uzyskać szczegółowe informacje na temat tego, jak to zrobić, należy zapoznać się z polem *ObjectRecOffset* opisanym w sekcji [“MQOD-deskryptor obiektu”](#) na stronie 479.

Nie można użyć więcej niż jednego z produktów *PutMsgRecOffset* i *PutMsgRecPtr*; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_PUT_MSG_RECORDS_ERROR, jeśli oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR)

Jest to adres pierwszego rekordu komunikatu umieszczonego w tabeli MQPMR. Opcji *PutMsgRecPtr* należy używać tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Można użyć opcji *PutMsgRecPtr* lub *PutMsgRecOffset*, aby określić rekordy umieszczania komunikatów, ale nie oba te rekordy; szczegółowe informacje na ten temat zawiera sekcja [“PutMsgRecOffset \(MQLONG\)”](#) na stronie 515. Jeśli produkt *PutMsgRecPtr* nie jest używany, ustaw go na pusty wskaźnik lub zerową liczbę bajtów.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtów o wartości all-null.

RecsPresent (MQLONG)

Jest to liczba rekordów komunikatów umieszczonych w tabeli MQPMR lub rekordów odpowiedzi MQRR, które zostały udostępnione przez aplikację. Liczba ta może być większa od zera tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Rekordy komunikatów i rekordy odpowiedzi są opcjonalne; aplikacja nie musi udostępniać żadnych rekordów lub może wybrać opcję udostępnienia rekordów tylko jednego typu. Jeśli jednak aplikacja udostępnia rekordy obu typów, musi ona udostępniać rekordy *RecsPresent* dla każdego typu.

Wartość *RecsPresent* nie musi być taka sama, jak liczba miejsc docelowych na liście dystrybucyjnej. Jeśli udostępniono zbyt wiele rekordów, przekroczenie tej wartości nie jest używane. Jeśli podano zbyt małą liczbę rekordów, dla właściwości komunikatu dla tych miejsc docelowych, które nie mają rekordów umieszczenia rekordów komunikatów (patrz *PutMsgRecOffset*), używane są wartości domyślne.

Jeśli wartość *RecsPresent* jest mniejsza od zera lub jest większa od zera, ale komunikat nie jest umieszczany na liście dystrybucyjnej, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RECS_PRESENT_ERROR.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

Nazwa ResolvedQMGr(MQCHAR48)

Jest to nazwa docelowego menedżera kolejek po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *ResolvedQName*, i może być nazwą lokalnego menedżera kolejek.

Jeśli *ResolvedQName* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, *ResolvedQMGrName* jest nazwą grupy współużytkowania kolejki. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, *ResolvedQName* może być nazwą grupy współużytkowania kolejki lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejki (rodzaj zwróconej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

To jest pole wyjściowe. Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

ResolvedQName (MQCHAR48)

Jest to nazwa kolejki docelowej po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa kolejki, która istnieje w menedżerze kolejek identyfikowanego przez produkt *ResolvedQMGrName*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

To jest pole wyjściowe. Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

ResponseRecPrzesunięcie (MQLONG)

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. *ResponseRecOffset* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Podczas umieszczania komunikatu na liście dystrybucyjnej można podać tablicę jednego lub większej liczby rekordów odpowiedzi MQRR, aby zidentyfikować kolejki, do których komunikat nie został pomyślnie wysłany (pole *CompCode* w MQRR), oraz przyczynę każdego niepowodzenia (pole *Reason* w tabeli MQRR). Być może komunikat nie został wysłany, ponieważ kolejka nie została otwarta lub operacja put nie powiodła się. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (oznacza to, że niektóre komunikaty zostały wysłane pomyślnie, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn); kod przyczyny MQRC_MULTIPLE_UZASADNIENIA wywołania wskazuje tę sprawę. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest zwracana w parametrze *Reason* wywołania MQPUT lub MQPUT1, a rekordy odpowiedzi nie są ustawione.

Zwykle istnieje wiele rekordów odpowiedzi, ponieważ istnieją rekordy obiektów określone przez MQOD, gdy lista dystrybucyjna jest otwierana; w razie potrzeby każdy rekord odpowiedzi jest ustawiany na kod zakończenia i kod przyczyny dla umieszczenia w kolejce identyfikowanej przez odpowiedni rekord obiektu. Kolejki z listy dystrybucyjnej, które nie otwierają się, muszą nadal mieć przypisane rekordy odpowiedzi dla odpowiednich pozycji w tablicy, chociaż są one ustawione na kod zakończenia i kod przyczyny wynikający z operacji otwarcia, a nie operacji put.

Liczba rekordów odpowiedzi może się różnić od liczby rekordów obiektów. Jeśli liczba rekordów odpowiedzi jest mniejsza niż rekordy obiektów, aplikacja może nie być w stanie zidentyfikować wszystkich miejsc docelowych, dla których operacja put nie powiodła się, lub przyczyny niepowodzeń. Jeśli istnieje więcej rekordów odpowiedzi niż rekordy obiektów, nadwyżka nie jest używana (choć nadal musi istnieć możliwość uzyskania dostępu do nich). Rekordy odpowiedzi są opcjonalne, ale jeśli są one podane, muszą być z nich *RecsPresent*.

Należy udostępnić rekordy odpowiedzi w podobny sposób do rekordów obiektów w tabeli MQOD, podając przesunięcie w składce *ResponseRecOffset* lub podając adres w programie *ResponseRecPtr*, aby uzyskać szczegółowe informacje na temat tego, jak to zrobić, należy zapoznać się z polem *ObjectRecOffset* opisanym w sekcji [“MQOD-deskryptor obiektu” na stronie 479](#). Należy jednak używać nie więcej niż jednego z produktów *ResponseRecOffset* i *ResponseRecPtr*; wywołanie nie powiodło się z kodem przyczyny MQRC_RESPONSE_RECORDS_ERROR, jeśli oba są niezerowe.

W przypadku wywołania MQPUT1 pole to musi być równe zero. Dzieje się tak dlatego, że informacje o odpowiedzi (jeśli są wymagane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

ResponseRecPtr (MQPTR)

Jest to adres pierwszego rekordu odpowiedzi MQRR. *ResponseRecPtr* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *RecsPresent* wynosi zero.

Użyj opcji *ResponseRecPtr* lub *ResponseRecOffset*, aby określić rekordy odpowiedzi, ale nie oba te rekordy; szczegółowe informacje na ten temat zawiera sekcja [“ResponseRecPrzesunięcie \(MQLONG\)” na stronie 516](#). Jeśli produkt *ResponseRecPtr* nie zostanie użyty, ustaw go na pusty wskaźnik lub bajty o wartości NULL.

W przypadku wywołania MQPUT1 pole to musi być pustym wskaźnikiem lub bajtami o wartości NULL. Dzieje się tak dlatego, że informacje o odpowiedzi (jeśli są wymagane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_2.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtów o wartości all-null.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQPMO_STRUC_ID

Identyfikator struktury opcji put-message.

Dla języka programowania C jest również zdefiniowana stała zmienna MQPMO_STRUC_ID_ARRAY; ma taką samą wartość jak MQPMO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQPMO_STRUC_ID.

Limit czasu (MQLONG)

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest -1.

Licznik UnknownDest(MQLONG)

Jest to liczba komunikatów, które bieżące wywołanie MQPUT lub MQPUT1 zostało pomyślnie wysłane do kolejek na liście dystrybucyjnej, które są rozstrzygane do kolejek zdalnych. Komunikaty, które menedżer kolejek zachowuje tymczasowo w formie listy dystrybucyjnej, są liczone jako liczba pojedynczych miejsc docelowych, które zawierają te listy dystrybucyjne. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *Version* jest mniejsza niż MQPMO_VERSION_1.

To pole jest niezdefiniowane w produkcie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

Wersja (MQLONG)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

MQPMO_VERSION_1

Struktura opcji komendy put-message w wersji Version-1 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQPMO_VERSION_2

Struktura opcji komendy put-message w wersji Version-2 .

Ta wersja jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.

MQPMO_VERSION_3

Struktura opcji komendy put-message w wersji Version-3 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól.

Następująca stała określa numer wersji bieżącej wersji:

MQPMO_CURRENT_VERSION

Bieżąca wersja struktury opcji put-message.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQPMO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQPMO

Tabela 68. Początkowe wartości pól w MQPMO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQPMO_STRUC_ID	'PMO~'
<i>Version</i>	MQPMO_VERSION_1	1
<i>Options</i>	MQPMO_BRAK	0

Tabela 68. Początkowe wartości pól w MQPMO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>Timeout</i>	Brak	-1
<i>Context</i>	Brak	0
<i>KnownDestCount</i>	Brak	0
<i>UnknownDestCount</i>	Brak	0
<i>InvalidDestCount</i>	Brak	0
<i>ResolvedQName</i>	Brak	Pusty łańcuch lub odstępy
<i>ResolvedQMgrName</i>	Brak	Pusty łańcuch lub odstępy
<i>RecsPresent</i>	Brak	0
<i>PutMsgRecFields</i>	MQPMRF_NONE	0
<i>PutMsgRecOffset</i>	Brak	0
<i>ResponseRecOffset</i>	Brak	0
<i>PutMsgRecPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>ResponseRecPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>OriginalMsgHandle</i>	MQHM_NONE	0
<i>NewMsgHandle</i>	MQHM_NONE	0
<i>Action</i>	MQACTP_NEW	0
<i>PubLevel</i>	Brak	9

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraParametr MQPMO_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of
                                MQPUT and MQPUT1 */
    MQLONG     Timeout;          /* Reserved */
    MQHOBJS    Context;          /* Object handle of input queue */
    MQLONG     KnownDestCount;   /* Number of messages sent
                                successfully to local queues */
    MQLONG     UnknownDestCount; /* Number of messages sent
                                successfully to remote queues */
    MQLONG     InvalidDestCount; /* Number of messages that could not
                                be sent */
}
```

```

MQCHAR48 ResolvedQName;      /* Resolved name of destination
                               queue */
MQCHAR48 ResolvedQMGrName;    /* Resolved name of destination queue
                               manager */
/* Ver:1 */
MQLONG RecsPresent;           /* Number of put message records or
                               response records present */
MQLONG PutMsgRecFields;       /* Flags indicating which MQPMR fields
                               are present */
MQLONG PutMsgRecOffset;       /* Offset of first put message record
                               from start of MQPMO */
MQLONG ResponseRecOffset;     /* Offset of first response record
                               from start of MQPMO */
MQPTR PutMsgRecPtr;           /* Address of first put message
                               record */
MQPTR ResponseRecPtr;         /* Address of first response record */
/* Ver:2 */
MQHMSG OriginalMsgHandle;     /* Original message handle */
MQHMSG NewMsgHandle;          /* New message handle */
MQLONG Action;                /* The action being performed */
MQLONG PubLevel;              /* Subscription level */
/* Ver:3 */
};

```

Deklaracja języka COBOL

```

** MQPMO structure
10 MQPMO.
** Structure identifier
15 MQPMO-STRUCID PIC X(4).
** Structure version number
15 MQPMO-VERSION PIC S9(9) BINARY.
** Options that control the action of MQPUT and MQPUT1
15 MQPMO-OPTIONS PIC S9(9) BINARY.
** Reserved
15 MQPMO-TIMEOUT PIC S9(9) BINARY.
** Object handle of input queue
15 MQPMO-CONTEXT PIC S9(9) BINARY.
** Number of messages sent successfully to local queues
15 MQPMO-KNOWNDESTCOUNT PIC S9(9) BINARY.
** Number of messages sent successfully to remote queues
15 MQPMO-UNKNOWNDESTCOUNT PIC S9(9) BINARY.
** Number of messages that could not be sent
15 MQPMO-INVALIDDESTCOUNT PIC S9(9) BINARY.
** Resolved name of destination queue
15 MQPMO-RESOLVEDQNAME PIC X(48).
** Resolved name of destination queue manager
15 MQPMO-RESOLVEDQMGRNAME PIC X(48).
** Number of put message records or response records present
15 MQPMO-RECSPRESENT PIC S9(9) BINARY.
** Flags indicating which MQPMR fields are present
15 MQPMO-PUTMSGRECFIELDS PIC S9(9) BINARY.
** Offset of first put message record from start of MQPMO
15 MQPMO-PUTMSGRECOFFSET PIC S9(9) BINARY.
** Offset of first response record from start of MQPMO
15 MQPMO-RESPONSERECOFFSET PIC S9(9) BINARY.
** Address of first put message record
15 MQPMO-PUTMSGRECPtr POINTER.
** Address of first response record
15 MQPMO-RESPONSERECPtr POINTER.
** Original message handle
15 MQPMO-ORIGINALMSGHANDLE PIC S9(18) BINARY.
** New message handle
15 MQPMO-NEWMSGHANDLE PIC S9(18) BINARY.
** The action being performed
15 MQPMO-ACTION PIC S9(9) BINARY.
** Publish level
15 MQPMO-PUBLEVEL PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQPMO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 Options fixed bin(31), /* Options that control the action

```

```

of MQPUT and MQPUT1 */
3 Timeout          fixed bin(31), /* Reserved */
3 Context          fixed bin(31), /* Object handle of input queue */
3 KnownDestCount   fixed bin(31), /* Number of messages sent
                                   successfully to local queues */
3 UnknownDestCount fixed bin(31), /* Number of messages sent
                                   successfully to remote queues */
3 InvalidDestCount fixed bin(31), /* Number of messages that could
                                   not be sent */
3 ResolvedQName     char(48),      /* Resolved name of destination
                                   queue */
3 ResolvedQMGrName  char(48),      /* Resolved name of destination
                                   queue manager */
3 RecsPresent       fixed bin(31), /* Number of put message records or
                                   response records present */
3 PutMsgRecFields   fixed bin(31), /* Flags indicating which MQPMR
                                   fields are present */
3 PutMsgRecOffset   fixed bin(31), /* Offset of first put message
                                   record from start of MQPMO */
3 ResponseRecOffset fixed bin(31), /* Offset of first response record
                                   from start of MQPMO */
3 PutMsgRecPtr      pointer,        /* Address of first put message
                                   record */
3 ResponseRecPtr    pointer,        /* Address of first response
                                   record */
3 OriginalMsgHandle fixed bin(63), /* Original message handle */
3 NewMsgHandle      fixed bin(63); /* New message handle */
3 Action            fixed bin(31); /* The action being performed */
3 PubLevel          fixed bin(31); /* Publish level */

```

Deklaracja High Level Assembler

```

MQPMO          DSECT
MQPMO_STRUCID  DS    CL4    Structure identifier
MQPMO_VERSION  DS    F      Structure version number
MQPMO_OPTIONS  DS    F      Options that control the action of
*               MQPUT and MQPUT1
MQPMO_TIMEOUT  DS    F      Reserved
MQPMO_CONTEXT  DS    F      Object handle of input queue
MQPMO_KNOWNDESTCOUNT DS    F      Number of messages sent successfully
*               to local queues
MQPMO_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)

```

Wizualna deklaracja podstawowa

```

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-Put-rekord komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 69. Pola w MQPMR		
Pole	Opis	Temat
<i>MsgId</i>	Identyfikator komunikatu	MsgId
<i>CorrelId</i>	Identyfikator korelacji	CorrelId
<i>GroupId</i>	Identyfikator grupy	GroupId
<i>Feedback</i>	Informacja zwrotna lub kod przyczyny	Opinie
<i>AccountingToken</i>	Token rozliczania	AccountingToken

Przegląd produktu MQPMR

Dostępność: klienci AIX, HP-UX, IBM i, Solaris, Linux, Windowsi IBM MQ połączone z tymi systemami.

Cel: Użyj struktury MQPMR w celu określenia różnych właściwości komunikatu dla pojedynczego miejsca docelowego podczas umieszczania komunikatu na liście dystrybucyjnej. MQPMR jest strukturą wejścia/wyjścia dla wywołań MQPUT i MQPUT1.

Zestaw znaków i kodowanie: Dane w tabeli MQPMR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniu MQPUT lub MQPUT1, można określić różne wartości dla każdej kolejki docelowej na liście dystrybucyjnej. Niektóre pola są tylko danymi wejściowymi, inne są wejściami/wyjściami.

Uwaga: Ta struktura jest nietypowa w tym, że nie ma ustalonego układu. Pola w tej strukturze są opcjonalne, a obecność lub nieobecność każdego pola jest wskazywana przez flagi w polu *PutMsgRecFields* w MQPMO. Pola, które są obecne w polu **muszą występować w następującej kolejności**:

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Nieobecne pola nie zajmują miejsca w rekordzie.

Ponieważ tabela MQPMR nie ma stałego układu, definicja nie jest dostępna w plikach nagłówkowych, COPY i INCLUDE dla obsługiwanych języków programowania. Programista aplikacji musi utworzyć deklarację zawierającą pola wymagane przez aplikację, a następnie ustawić flagi w programie *PutMsgRecFields*, aby wskazać obecne pola.

Pola dla MQPMR

Struktura MQPMR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

AccountingToken (MQBYTE32)

Jest to znacznik rozliczeniowy, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1. Jest on przetwarzany w ten sam sposób, co pole *AccountingToken* w strukturze MQMD dla umieszczenia w jednej kolejce. Informacje na temat zawartości tego pola można znaleźć w opisie produktu *AccountingToken* w produkcie [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

To jest pole wejściowe.

CorrelId (MQBYTE24)

Jest to identyfikator korelacji, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1. Jest on przetwarzany w ten sam sposób, co pole *CorrelId* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowe, wartość w strukturze MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *CorrelId*.

Jeśli określono wartość MQPMO_NEW_CORREL_ID, *pojedynczy* nowy identyfikator korelacji jest generowany i używany dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania identyfikatora MQPMO_NEW_MSG_ID (patrz pole *MsgId*).

Jest to pole wejściowe/wyjściowe.

Opinia (MQLONG)

Jest to kod informacji zwrotnej, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1. Jest on przetwarzany w ten sam sposób, co pole *Feedback* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

To jest pole wejściowe.

GroupId (MQBYTE24)

GroupId to identyfikator grupy, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1. Jest on przetwarzany w ten sam sposób, co pole *GroupId* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowe, wartość w strukturze MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *GroupId*. Wartość jest przetwarzana zgodnie z opisem w sekcji [Kolejność fizyczna w kolejce](#), ale z następującymi różnicami:

- Element `GroupId` jest tworzony na podstawie wartości `QMName` i znacznika czasu. Dlatego też, aby zachować unikalny identyfikator grupy `GroupId`, należy zachować unikalne nazwy menedżerów kolejek. Nie należy również ustawiać zegarów na komputerze z menedżerami kolejek.
- W tych przypadkach, w których zostanie użyty nowy identyfikator grupy, menedżer kolejek generuje inny identyfikator grupy dla każdego miejsca docelowego (to znaczy nie ma dwóch miejsc docelowych o tym samym identyfikatorze grupy).
- W tych przypadkach, w których wartość w tym polu będzie używana, wywołanie kończy się niepowodzeniem z kodem przyczyny `MQRC_GROUP_ID_ERROR`

Jest to pole wejściowe/wyjściowe.

MsgId (MQBYTE24)

Jest to identyfikator komunikatu, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur `MQOR` udostępnionych w wywołaniu `MQOPEN` lub `MQPUT1`. Jest on przetwarzany w ten sam sposób, co pole `MsgId` w strukturze `MQMD` dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie `MQPMR` lub istnieje mniej rekordów `MQPMR` niż miejsc docelowe, wartość w strukturze `MQMD` jest używana dla tych miejsc docelowych, które nie mają rekordu `MQPMR` zawierającego pole `MsgId`. Jeśli ta wartość to `MQMI_NONE`, dla *każdego* tych miejsc docelowych jest generowany nowy identyfikator komunikatu (co oznacza, że żadne dwa z tych miejsc docelowych nie mają takiego samego identyfikatora komunikatu).

Jeśli określono wartość `MQPMO_NEW_MSG_ID`, nowe identyfikatory komunikatów są generowane dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy `MQPMR`. Różni się to od sposobu przetwarzania tabeli `MQPMO_NEW_CORREL_ID` (patrz pole *CorrelId*).

Jest to pole wejściowe/wyjściowe.

Wartości początkowe i deklaracje języka dla MQPMR

Dla tej struktury nie zdefiniowano wartości początkowych, ponieważ w nagłówkach, plikach `COPY` i `INCLUDE` dla obsługiwanych języków programowania nie są udostępniane żadne deklaracje struktury. Przykładowe deklaracje pokazują, jak zadeklarować strukturę, jeśli wszystkie pola są wymagane.

Deklaracja C

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
    MQBYTE24  MsgId;           /* Message identifier */
    MQBYTE24  CorrelId;        /* Correlation identifier */
    MQBYTE24  GroupId;         /* Group identifier */
    MQLONG    Feedback;        /* Feedback or reason code */
    MQBYTE32  AccountingToken; /* Accounting token */
};
```

Deklaracja języka COBOL

```
**      MQPMR structure
10 MQPMR.
**      Message identifier
15 MQPMR-MSGID          PIC X(24).
**      Correlation identifier
15 MQPMR-CORRELID       PIC X(24).
**      Group identifier
15 MQPMR-GROUPID        PIC X(24).
**      Feedback or reason code
15 MQPMR-FEEDBACK       PIC S9(9) BINARY.
**      Accounting token
15 MQPMR-ACCOUNTINGTOKEN PIC X(32).
```

Deklaracja PL/I


```

dcl
1 MQPMR based,
3 MsgId          char(24),      /* Message identifier */
3 CorrelId       char(24),      /* Correlation identifier */
3 GroupId        char(24),      /* Group identifier */
3 Feedback       fixed bin(31), /* Feedback or reason code */
3 AccountingToken char(32);     /* Accounting token */

```

Wizualna deklaracja podstawowa

```

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-nagłówek reguł i formatowania

W tej sekcji opisano reguły i nagłówki formatowania, jakie pola zawiera, a także wartości początkowe tych pól.

Przegląd produktu MQRFH

Dostępność: Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Cel: Struktura MQRFH definiuje układ nagłówka reguł i formatowania. Ten nagłówek służy do wysyłania danych łańcuchowych w postaci par nazwa/wartość.

Nazwa formatu: MQFMT_RF_HEADER.

Zestaw znaków i kodowanie: pola w strukturze MQRFH (w tym *NameValueString*) znajdują się w zestawie znaków i kodowaniu podanym w polach *CodedCharSetId* i *Encoding* w strukturze nagłówka poprzedzającym MQRFH lub przez te pola w strukturze MQMD, jeśli wartość MQRFH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Pola dla MQRFH

Struktura MQRFH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CodedCharSetId (MQLONG)

Określa identyfikator zestawu znaków dla danych, które są następujące *NameValueString*; nie ma zastosowania do danych znakowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie można użyć tabeli MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcyjce MQMD jest MQAT_BROKER.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Kodowanie (MQLONG)

Określa kodowanie liczbowe dla danych, które są następujące *NameValueString* ; nie ma zastosowania do danych liczbowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

Flagi (MQLONG)

Można określić następujące elementy:

MQRFH_NONE

Brak flag.

Wartością początkową tego pola jest MQRFH_NONE.

Format (MQCHAR8)

Określa nazwę formatu danych, które są następujące *NameValueString*.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

NameValueString (MQCHARn)

Jest to łańcuch znaków o zmiennej długości zawierający pary nazwa/wartość w postaci:

```
name1 value1 name2 value2 name3 value3 ...
```

Każda nazwa lub wartość musi być oddzielona od przylegającej nazwy lub wartości przez jeden lub więcej znaków odstępu; te odstępy nie są znaczące. Nazwa lub wartość może zawierać spacje, poprzedzając je przedrostkiem i przyrostem nazwy lub wartości znakami podwójnego cudzysłowu. Wszystkie znaki między otwieranym znakiem podwójnego cudzysłowu a pasującym znakiem podwójnego cudzysłowu zamykającego są traktowane jako znaczące. W poniższym przykładzie nazwą jest FAMOUS_WORDS, a wartością jest Hello World:

```
FAMOUS_WORDS "Hello World"
```

Nazwa lub wartość może zawierać dowolne znaki inne niż znak o kodzie zero (który działa jako ogranicznik dla produktu *NameValueString*). Jednak w celu ułatwienia współdziałania aplikacja może ograniczyć nazwy do następujących znaków:

- Pierwszy znak: wielkie lub małe litery (od A do Z, lub od a do z) lub podkreślenie.
- Kolejne znaki: wielkie lub małe litery, cyfry dziesiętne (od 0 do 9), podkreślenie, myślnik lub kropka.

Jeśli nazwa lub wartość zawiera jeden lub kilka podwójnych cudzysłowów, nazwa lub wartość muszą być ujęte w znaki podwójnego cudzysłowu, a każdy podwójny cudzysłów wewnątrz łańcucha musi być podwójony:

```
Famous_Words "The program displayed ""Hello World"""
```

W nazwach i wartościach rozróżniana jest wielkość liter, oznacza to, że małe litery nie są traktowane tak samo, jak wielkie litery. Na przykład: FAMOUS_WORDS i Famous_Words to dwie różne nazwy.

Długość (w bajtach) *NameValueString* jest równa wartości *StrucLength* minus MQRFH_STRUC_LENGTH_FIXED. Aby uniknąć problemów z przekształcaniu danych użytkownika w niektórych środowiskach, należy zmienić tę długość na wielokrotność liczby czterech. Dopełniaj *NameValueString* odstępami do tej długości lub przerwij je wcześniej, umieszczając znak o kodzie zero następującym po ostatnim znaczącym znaku w łańcuchu. Znak o kodzie zero i bajty następujące po nim, aż do określonej długości *NameValueString*, są ignorowane.

Uwaga: Ponieważ długość tego pola nie jest ustalona, to pole jest pomijane w deklaracjach struktury, które są udostępniane dla obsługiwanych języków programowania.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQRFH_STRUC_ID

Identyfikator regułu i struktury nagłówka formatowania.

W przypadku języka programowania C jest również zdefiniowana stała `MQRFH_STRUC_ID_ARRAY`. Ma ona taką samą wartość jak `MQRFH_STRUC_ID`, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to `MQRFH_STRUC_ID`.

StrucLength (MQLONG)

Jest to długość (w bajtach) struktury `MQRFH`, w tym pole *NameValueString* na końcu struktury. Długość nie obejmuje żadnych danych użytkownika, które są następujące po polu *NameValueString*.

Aby uniknąć problemów z przekształcaniu danych użytkownika w niektórych środowiskach, produkt *StrucLength* musi być wielokrotnością liczby czterech.

Następująca stała określa długość części stałej struktury, to znaczy długość, z wyłączeniem pola *NameValueString*:

MQRFH_STRUC_LENGTH_FIXED

Długość stałej części struktury `MQRFH`.

Początkowa wartość tego pola to `MQRFH_STRUC_LENGTH_FIXED`.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQRFH_VERSION_1

Reguły Version-1 i struktura nagłówka formatowania.

Początkowa wartość tego pola to `MQRFH_VERSION_1`.

Wartości początkowe i deklaracje języka dla MQRFH

Tabela 70. Początkowe wartości pól w MQRFH dla MQRFH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	<code>MQRFH_STRUC_ID</code>	'RFH↵'
<i>Version</i>	<code>MQRFH_VERSION_1</code>	1
<i>StrucLength</i>	<code>MQRFH_STRUC_LENGTH_FIXED</code>	32
<i>Encoding</i>	<code>MQENC_NATIVE</code>	Zależy od środowiska
<i>CodedCharSetId</i>	<code>MQCCSI_UNDEFINED</code>	0
<i>Format</i>	<code>MQFMT_NONE</code>	Puste
<i>Flags</i>	<code>MQRFH_NONE</code>	0

Tabela 70. Początkowe wartości pól w MQRFH dla MQRFH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: 1. Symbol ~ reprezentuje pojedynczy pusty znak. 2. W języku programowania C: zmienna makraWartość MQRFH_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQRFH MyRFH = {MQRFH_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
    MQCHAR4 StrucId;          /* Structure identifier */
    MQLONG  Version;          /* Structure version number */
    MQLONG  StrucLength;       /* Total length of MQRFH including
                               NameValueString */
    MQLONG  Encoding;         /* Numeric encoding of data that follows
                               NameValueString */
    MQLONG  CodedCharSetId;    /* Character set identifier of data that
                               follows NameValueString */
    MQCHAR8 Format;           /* Format name of data that follows
                               NameValueString */
    MQLONG  Flags;            /* Flags */
};
```

Deklaracja języka COBOL

```
** MQRFH structure
10 MQRFH.
** Structure identifier
15 MQRFH-STRUCID PIC X(4).
** Structure version number
15 MQRFH-VERSION PIC S9(9) BINARY.
** Total length of MQRFH including NAMEVALUESTRING
15 MQRFH-STRUCLNGTH PIC S9(9) BINARY.
** Numeric encoding of data that follows NAMEVALUESTRING
15 MQRFH-ENCODING PIC S9(9) BINARY.
** Character set identifier of data that follows NAMEVALUESTRING
15 MQRFH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows NAMEVALUESTRING
15 MQRFH-FORMAT PIC X(8).
** Flags
15 MQRFH-FLAGS PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQRFH based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 StrucLength fixed bin(31), /* Total length of MQRFH including
                               NameValueString */
3 Encoding fixed bin(31), /* Numeric encoding of data that
                               follows NameValueString */
3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                               that follows NameValueString */
3 Format char(8), /* Format name of data that follows
                               NameValueString */
3 Flags fixed bin(31); /* Flags */
```

Deklaracja High Level Assembler

MQRFH	DSECT		
MQRFH_STRUCID	DS	CL4	Structure identifier
MQRFH_VERSION	DS	F	Structure version number
MQRFH_STRUCLength	DS	F	Total length of MQRFH including
*			NAMEVALUESTRING
MQRFH_ENCODING	DS	F	Numeric encoding of data that follows
*			NAMEVALUESTRING
MQRFH_CODEDCHARSETID	DS	F	Character set identifier of data that
*			follows NAMEVALUESTRING
MQRFH_FORMAT	DS	CL8	Format name of data that follows
*			NAMEVALUESTRING
MQRFH_FLAGS	DS	F	Flags
*			
MQRFH_LENGTH	EQU	*-MQRFH	
	ORG	MQRFH	
MQRFH_AREA	DS	CL(MQRFH_LENGTH)	

Wizualna deklaracja podstawowa

```

Type MQRFH
  StrucId      As String*4 'Structure identifier'
  Version      As Long     'Structure version number'
  StrucLength  As Long     'Total length of MQRFH including'
                        'NameValueString'
  Encoding     As Long     'Numeric encoding of data that follows'
                        'NameValueString'
  CodedCharSetId As Long   'Character set identifier of data that'
                        'follows NameValueString'
  Format       As String*8 'Format name of data that follows'
                        'NameValueString'
  Flags       As Long     'Flags'
End Type

```

MQRFH2 -reguły i nagłówki formatowania 2

W tej sekcji opisano reguły i nagłówki formatowania 2, jakie pola zawiera, a także wartości początkowe tych pól.

Przeгляд produktu MQRFH2

Dostępność

Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Przeznaczenie

Nagłówek MQRFH2 jest oparty na nagłówku MQRFH, ale umożliwia transport łańcuchów Unicode bez tłumaczenia, a także może przenosić liczbowe typy danych.

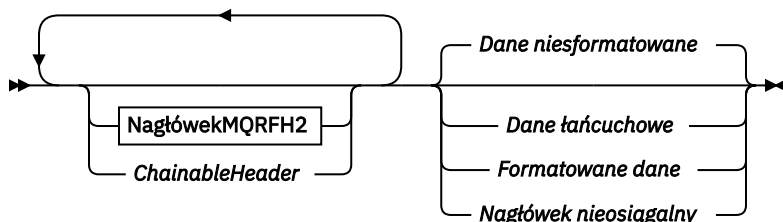
Struktura MQRFH2 definiuje format reguł i nagłówków formatowania version-2. Ten nagłówek służy do wysyłania danych, które zostały zakodowane przy użyciu składni typu XML. Komunikat może zawierać dwie lub więcej struktur MQRFH2 z serii, a dane użytkownika są opcjonalnie następujące po ostatniej strukturze produktu MQRFH2 w serii.

Nazwa formatu

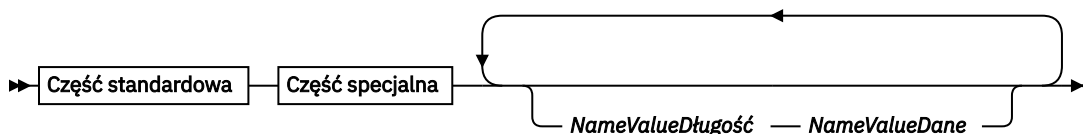
MQFMT_RF_HEADER_2

Składnia

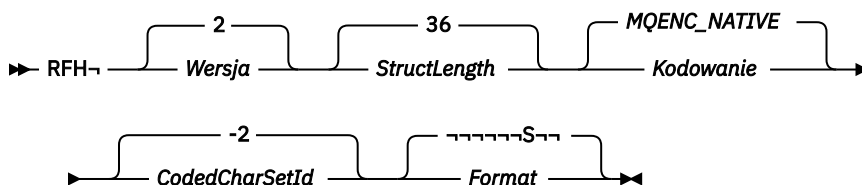
IBM MQ Komunikat



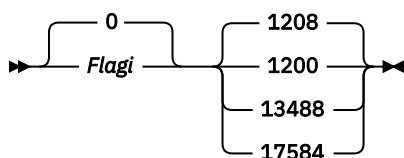
Nagłówek MQRFH2



Część standardowa



Część specjalna



Zestaw znaków i kodowanie

Reguły specjalne mają zastosowanie do zestawu znaków i kodowania używanego w strukturze MQRFH2 :

- Pola inne niż *NameValueData* znajdują się w zestawie znaków i kodowaniu podanym w polach *CodedCharSetId* i *Encoding* w strukturze nagłówka, które poprzedzają MQRFH2, lub według tych pól w strukturze MQMD , jeśli MQRFH2 znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Jeśli w wywołaniu MQGET jest określona wartość MQGMO_CONVERT , menedżer kolejek przekształca pola MQRFH2 inne niż *NameValueData* na żądany zestaw znaków i kodowanie.

- *NameValueData* znajduje się w zestawie znaków podanym w polu *NameValueCCSID* . Tylko wymienione zestawy znaków Unicode są poprawne dla produktu *NameValueCCSID* . Szczegółowe informacje zawiera opis produktu *NameValueCCSID* .

Niektóre zestawy znaków mają reprezentację, która zależy od kodowania. Jeśli *NameValueCCSID* jest jednym z tych zestawów znaków, *NameValueData* musi być w tym samym kodowaniu, co inne pola w MQRFH2.

Gdy opcja MQGMO_CONVERT jest określona w wywołaniu MQGET , menedżer kolejek przekształca *NameValueData* w żądane kodowanie, ale nie zmienia jego zestawu znaków.

Pola dla MQRFH2

Struktura MQRFH2 zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CodedCharSetId (MQLONG)

Określa identyfikator zestawu znaków dla danych, które są następujące po ostatnim polu *NameValueData* . Nie ma on zastosowania do danych znakowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych po tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie można użyć tabeli MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcyjce MQMD jest MQAT_BROKER.

Wartością początkową tego pola jest MQCCSI_INHERIT.

Kodowanie (MQLONG)

Określa kodowanie liczbowe dla danych, które są zgodne z ostatnim polem *NameValueData* ; nie ma ono zastosowania do danych liczbowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

Flagi (MQLONG)

Wartością początkową tego pola jest MQRFH_NONE. MQRFH_NONE należy podać.

MQRFH_NONE

Brak flag.

MQRFH_INTERNAL

Nagłówek MQRFH2 zawiera właściwości zestawu wewnętrznego.

MQRFH_INTERNAL jest przeznaczony do użycia przez menedżera kolejek.

Górne 16 bitów, MQRFH_FLAGS_RESTRICTED_MASK, są zarezerwowane dla flag zestawów menedżerów kolejek. Flagi, które mogą być ustawione przez użytkownika, są zdefiniowane w dolnych 16 bitach.

Format (MQCHAR8)

Określa nazwę formatu danych, które są zgodne z ostatnim polem *NameValueData* .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

NameValueCCSID (MQLONG)

Ten parametr określa identyfikator kodowanego zestawu znaków dla danych w polu *NameValueData* . Różni się on od zestawu znaków innych łańcuchów w strukturze MQRFH2 i może różnić się od zestawu znaków danych (jeśli istnieją), które są następujące po ostatnim polu *NameValueData* na końcu struktury.

NameValueCCSID musi mieć jedną z następujących wartości:

CCSID	Znaczenie
1200	Otwarto UCS-2
13488	Podzbiór UCS-2 2.0
17584	Podzbiór UCS-2 2.1 (zawiera symbol Euro)

CCSID	Znaczenie
1208	UTF-8

W przypadku zestawów znaków UCS-2 kodowanie (kolejność bajtów) *NameValueData* musi być takie samo, jak kodowanie innych pól w strukturze MQRFH2. Znaki zastępcze (X'D800'przez X'DFFF') nie są obsługiwane.

Uwaga: Jeśli program *NameValueCCSID* nie ma jednej z wymienionych powyżej wartości, a struktura MQRFH2 wymaga konwersji w wywołaniu MQGET, wywołanie kończy się kodem przyczyny MQRC_SOURCE_CCSID_ERROR, a komunikat jest zwracany bez konwersji.

Wartość początkowa tego pola to 1208.

Dane NameValue(MQCHARn)

NameValueData to pole o zmiennej długości, które zawiera folder zawierający pary nazwa/wartość właściwości komunikatu. Folder jest łańcuchem znakowym o zmiennej długości zawierającym dane zakodowane przy użyciu składni typu XML. Długość łańcucha znaków w bajtach jest podana w polu *NameValueLength*, które poprzedza pole *NameValueData*. Długość musi być wielokrotnością czterech.

Pola *NameValueLength* i *NameValueData* są opcjonalne, ale jeśli występują, muszą one występować jako para i być sąsiadującymi. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

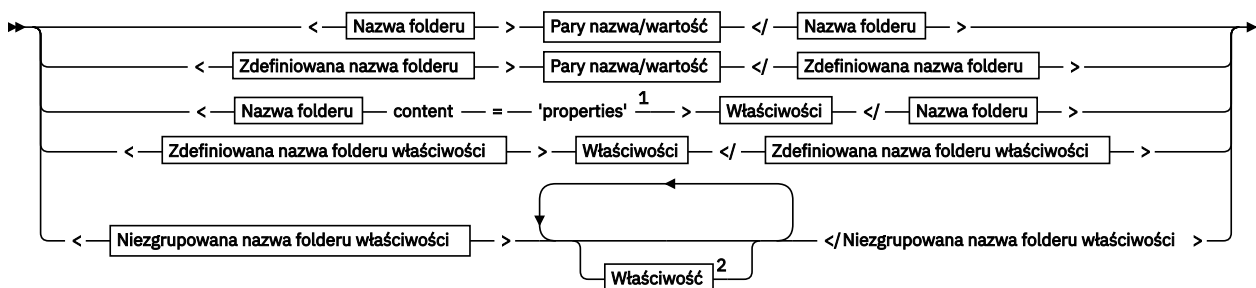
```
length1 data1 length2 data2 length3 data3
```

Produkt *NameValueData* nie jest przekształcany w zestaw znaków określony w wywołaniu MQGET. Nawet jeśli komunikat jest pobierany za pomocą opcji MQGMO_CONVERT w działaniu *NameValueData*, pozostaje w oryginalnym zestawie znaków. Jednak *NameValueData* jest konwertowane na kodowanie określone w wywołaniu MQGET.

Uwaga:

- Ponieważ pola te są opcjonalne, są one pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.
- Terminy "zdefiniowane" i "zastrzeżone" są używane w diagramie składni. "Zdefiniowane" oznacza, że nazwa jest używana przez produkt IBM MQ. "Zarezerwowane" oznacza, że nazwa jest zarezerwowana do użycia w przyszłości przez produkt IBM MQ.

NameValueData Składnia



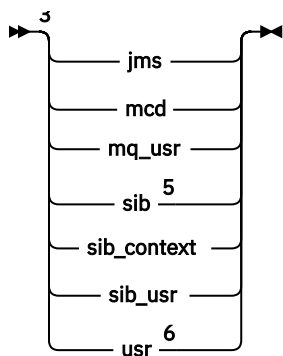
Nazwa folderu

```
>> Nazwa <<
```

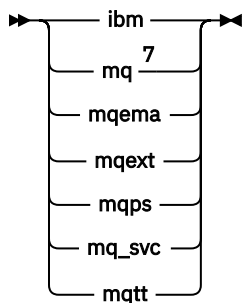
Zdefiniowana nazwa folderu

```
>>
   |
   +-- psc -->
   |
   +-- pscr -->
```

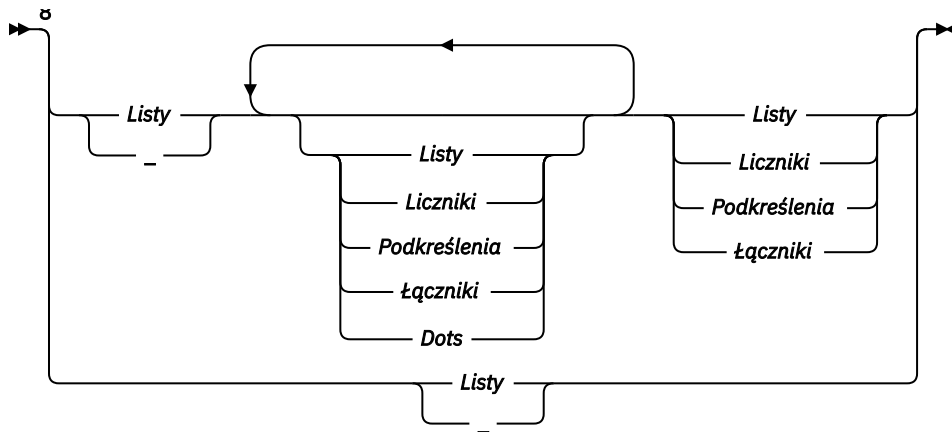
Zdefiniowana nazwa folderu właściwości



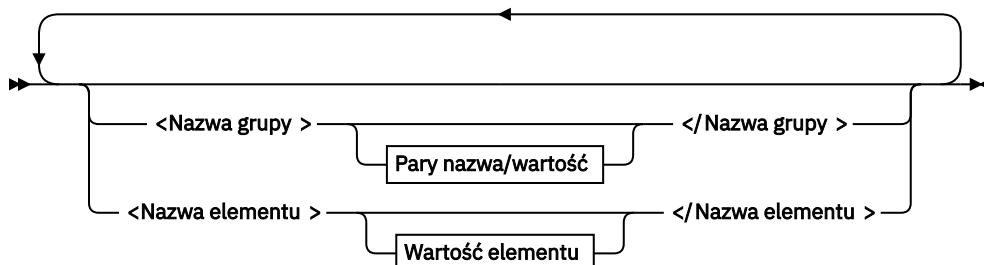
Niezgrupowana nazwa folderu właściwości



Nazwa



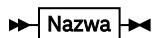
Pary nazwa/wartość



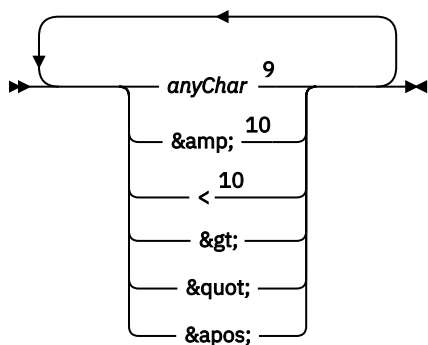
Nazwa grupy



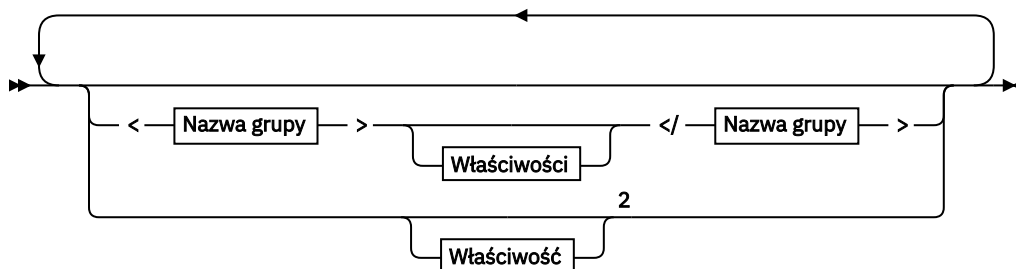
Nazwa elementu



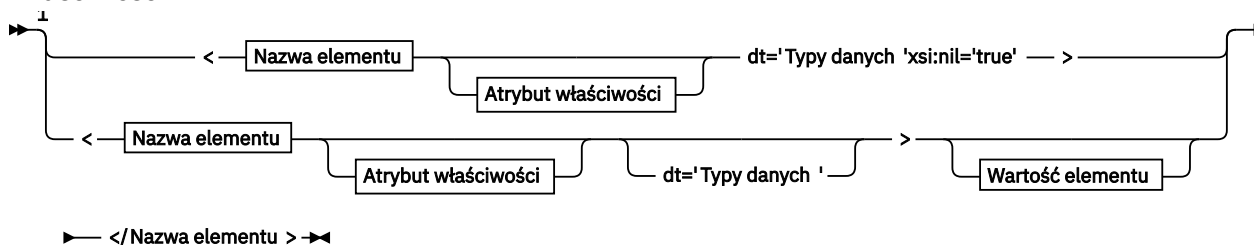
Wartość elementu



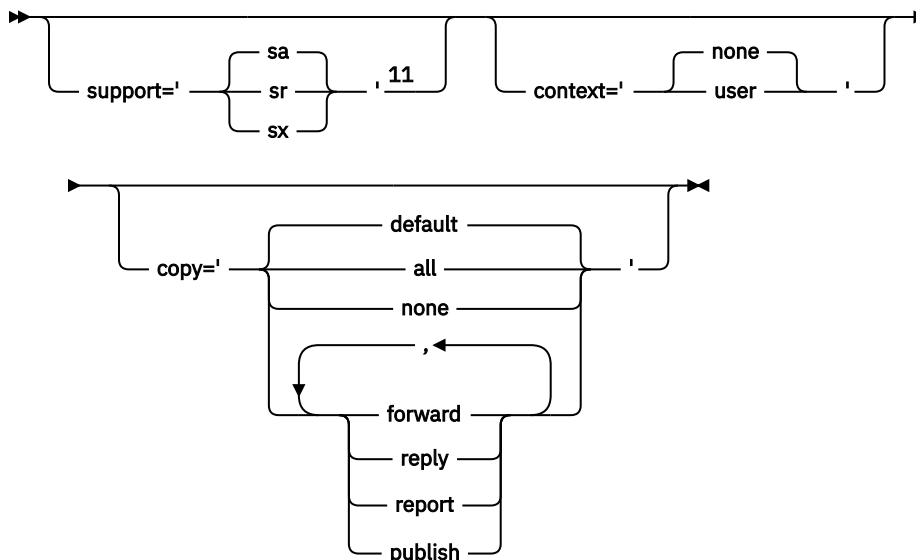
Właściwości



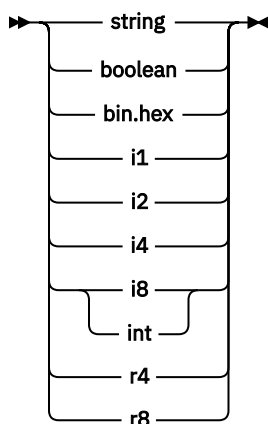
Właściwość



Atrybut właściwości



Typy danych



Uwagi:

- ¹ Poprawne są podwójne cudzysłowy lub pojedyncze cudzysłowy.
- ² Nie używaj niepoprawnej nazwy właściwości; patrz [“Niepoprawna nazwa właściwości”](#) na stronie 546. Zastrzeżonej nazwy właściwości należy używać tylko dla jego zdefiniowanego celu. Patrz [“Zdefiniowane nazwy właściwości”](#) na stronie 546.
- ³ Nazwa musi być zapisana małymi literami.
- ⁴ Obsługiwany jest tylko jeden folder psc i psc:r .
- ⁵ Tylko właściwości w pierwszym nagłówku MQRFH2 są znaczące. Usługa WebSphere Application Server Service Integration Bus ignoruje foldery sib, sib_context i sib_usr w kolejnych nagłówkach produktu MQRFH2 .
- ⁶ Nie więcej niż jeden folder usr musi być obecny w MQRFH2. Właściwości w folderze usr muszą wystąpić nie więcej niż jeden raz.
- ⁷ Istotne są tylko właściwości w pierwszym folderze mq . Jeśli folder ma wartość UTF-8, obsługiwane są tylko znaki jednobajtowe UTF-8 . Jedynym białym znakiem jest Unicode U+0020.
- ⁸ Poprawne znaki są zdefiniowane w specyfikacji XML W3C i składają się głównie z kategorii Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, i Nd ; patrz [Kategorie znaków Unicode](#).
- ⁹ Wszystkie znaki są znaczące. Odstępy początkowe i końcowe są częścią wartości elementu.
- ¹⁰ Nie należy używać niepoprawnego znaku. Patrz [“Nieprawidłowe znaki”](#) na stronie 546. Należy użyć sekwencji zmiany znaczenia, a nie tych niepoprawnych znaków.
- ¹¹ Atrybut właściwości obsługi jest poprawny tylko w folderze mq .

Nazwa folderu

NameValueData zawiera pojedynczy folder. Aby utworzyć wiele folderów, utwórz wiele pól *NameValueData* . Istnieje możliwość utworzenia wielu pól *NameValueData* w jednym nagłówku MQRFH2 w obrębie komunikatu. Alternatywnie można utworzyć wiele połączonych nagłówków MQRFH2 , z których każda zawiera wiele pól *NameValueData* .

Kolejność nagłówków MQRFH2 i kolejność pól *NameValueData* nie powodują różnic w zawartości logicznej folderu. Jeśli ten sam folder jest obecny więcej niż jeden raz w komunikacie, folder jest analizowany jako całość. Jeśli ta sama właściwość występuje w przypadku wielu instancji tego samego folderu, jest ona analizowana jako lista.

Alternatywne sposoby fizycznego przechowywania folderu w komunikacie nie mają wpływu na poprawną analizę składni MQRFH2 .

Cztery foldery nie są zgodne z tą regułą. Analizowana jest tylko pierwsza instancja folderu mq, sib, sib_context i sib_usr .

Jeśli ta sama właściwość występuje więcej niż jeden raz w połączonej treści połączonych nagłówków MQRFH2 , zostanie przeanalizowana tylko pierwsza instancja tej właściwości. Jeśli właściwość jest ustawiona za pomocą wywołania interfejsu API, takiego jak MQSETMP, i jest dodawana do MQRFH2 bezpośrednio przez aplikację, to wywołanie API ma pierwszeństwo.

Nazwa folderu to nazwa folderu zawierającego pary nazwa/wartość lub grupy. Grupy i pary nazwa/wartość mogą być mieszane na tym samym poziomie w drzewie folderów; patrz [Rysunek 1 na stronie 536](#). Nie należy łączyć nazwy grupy i nazwy elementu; patrz [Rysunek 2 na stronie 536](#)

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>  
<group3><nvp1> value </nvp1></group3><nvp3> value </nvp3>
```

Rysunek 1. Prawidłowe zastosowania grup i par nazwa/wartość

```
<group1><nvp1>value</nvp1>value</group1>
```

Rysunek 2. Niepoprawne użycie grup i par nazwa/wartość

Nie należy używać niepoprawnej nazwy folderu lub zastrzeżonej nazwy folderu; patrz [“Niepoprawna nazwa ścieżki” na stronie 546](#) i [“Zastrzeżony folder lub nazwa folderu właściwości” na stronie 545](#). Należy użyć zdefiniowanej nazwy folderu tylko dla jego zdefiniowanego celu; patrz [“Zdefiniowana nazwa folderu” na stronie 537](#).

Jeśli atrybut 'content=properties' zostanie dodany do znacznika nazwy folderu, folder stanie się folderem właściwości; patrz [Rysunek 3 na stronie 536](#).

```
<myFolder></myFolder>  
<myPropertyFolder content='properties'></myPropertyFolder>
```

Rysunek 3. Przykład folderu i folderu właściwości

W nazwach folderów rozróżniana jest wielkość liter. Nazwy folderów i nazwy folderów właściwości współużytkują tę samą przestrzeń nazw. Muszą mieć różne nazwy. Folder1 w produkcie [Rysunek 4 na stronie 536](#) musi być inną nazwą niż Folder2 w [Rysunek 5 na stronie 536](#).

```
<Folder1><NVP1>value</NVP1></Folder1>
```

Rysunek 4. Folder1 przestrzeń nazw

```
<Folder2 content='properties'><Property1>value</Property1></Folder2>
```

Rysunek 5. Folder2 przestrzeń nazw

Grupy, właściwości i pary nazwa/wartość w różnych folderach mają różne przestrzenie nazw. Property1 w produkcie [Rysunek 5 na stronie 536](#) jest inną właściwością niż Property1 w produkcie [Rysunek 6 na stronie 536](#).

```
<Folder3 content='properties'><Property1>value</Property1></Folder3>
```

Rysunek 6. Folder3 przestrzeń nazw

Foldery właściwości są różne dla folderów innych niż właściwości w dwóch ważnych aspektach:

1. Foldery właściwości zawierają właściwości, a foldery bez właściwości zawierają pary nazwa/wartość. Foldery różnią się nieznacznie, składniowo.

2. Aby uzyskać dostęp do właściwości komunikatów, należy użyć zdefiniowanych interfejsów, takich jak właściwości MQI właściwości lub właściwości komunikatu produktu JMS . Interfejsy zapewniają, że foldery właściwości w MQRFH2 są poprawnie sformatowane. Poprawnie sformatowany folder właściwości jest interoperacyjny między menedżerami kolejek na różnych platformach i różnych wersjach.

Właściwość komunikatu MQI jest odpornym sposobem na odczytywanie i zapisywanie MQRFH2, a także umożliwia uniknięcie trudności związanych z poprawnym analizowaniem MQRFH2 .

Zdefiniowana nazwa folderu

Zdefiniowana nazwa folderu to nazwa folderu, który jest zastrzeżony do użycia przez produkt IBM MQ lub inny produkt. Nie należy tworzyć folderu o tej samej nazwie i nie dodawać własnych par nazwa/wartość do folderów. Zdefiniowane foldery to psc i pscrr .

psc i pscrr są używane w kolejce publikowania/subskrypcji.

Posegmentowany komunikat umieszczony na MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED nie może zawierać MQRFH2 o zdefiniowanej nazwie folderu. MQPUT nie powiodło się z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Zdefiniowana nazwa folderu właściwości

Zdefiniowana nazwa folderu właściwości to nazwa folderu właściwości używanego przez produkt IBM MQ lub inny produkt. Informacje o nazwach folderów i ich treści zawiera sekcja [Foldery właściwości](#). Zdefiniowane nazwy folderów właściwości są podzbiorem wszystkich nazw folderów zarezerwowanych przez program IBM MQ; patrz [“Zastrzeżony folder lub nazwa folderu właściwości”](#) na stronie 545.

Każdy element zapisany w zdefiniowanym folderze właściwości jest właściwością. Element zapisany w zdefiniowanym folderze właściwości nie może mieć atrybutu content='properties' .

Właściwości można dodawać tylko do zdefiniowanych folderów właściwości usr , mq_usr i sib_usr . W innych folderach właściwości, takich jak mq i sib , produkt IBM MQ ignoruje lub odrzuca właściwości, których nie rozpoznaje.

W opisie każdego zdefiniowanego folderu właściwości znajduje się lista właściwości zdefiniowanych przez produkt IBM MQ , które mogą być używane przez aplikacje. Dostęp do niektórych właściwości można uzyskać pośrednio poprzez ustawienie lub uzyskanie właściwości JMS , a niektóre z nich są dostępne bezpośrednio przy użyciu wywołań MQI produktu MQSETMP i MQINQMP .

Zdefiniowane foldery właściwości zawierają również inne właściwości, które produkt IBM MQ zarezerwował, ale które aplikacje nie mają dostępu do tych właściwości. Nazwy zarezerwowanych właściwości nie są wyświetlane na liście. W folderach właściwości usr , mq_usr i sib_usr nie są dostępne żadne właściwości zastrzeżone. Nie należy jednak tworzyć właściwości z niepoprawnymi nazwami właściwości; patrz [“Niepoprawna nazwa właściwości”](#) na stronie 546.

Foldery właściwości

jms

Folder jms zawiera pola nagłówka JMS i właściwości JMSX, które nie mogą zostać w pełni wyrażone w deskrypcorze MQMD. Folder jms zawsze znajduje się w nagłówku MQRFH2 JMS.

Tabela 71. jms - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
JMSDestination	jms.Dst	string	<jms><Dst>destination</Dst></jms>

Tabela 71. <i>jms</i> - nazwa właściwości, synonim, typ danych i folder (kontynuacja)			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
JMSExpiration	.jms.Expiration	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>

Nie należy dodawać własnych właściwości w folderze *jms*.

mcd

Folder *mcd* zawiera właściwości opisujące format komunikatu. Na przykład właściwość domeny usługi komunikatu *Msd* identyfikuje komunikat JMS jako *JMSTextMessage*, *JMSBytesMessage*, *JMSStreamMessage*, *JMSMapMessage*, *JMSObjectMessage* lub wartość *NULL*.

Folder *mcd* jest zawsze obecny w komunikacie JMS zawierającym nagłówek *MQRFH2*.

Jest on zawsze obecny w komunikacie zawierającym nagłówek *MQRFH2* wysłany z produktu IBM Integration Bus. Opisuje on domenę, format, typ i zestaw komunikatu.

Tabela 72. Synonim, typ danych, folder i nazwa właściwości <i>mcd</i>			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	mcd.Msd	string	<mcd><Msd>messageDomain</Msd></mcd>
	mcd.Set	string	<mcd><Set>messageDomain</Set></mcd>
	mcd.Type	string	<mcd><Type>messageDomain</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>messageDomain</Fmt></mcd>

Nie należy dodawać własnych właściwości w folderze *mcd*.

mq_usr

Produkt mq_usr zawiera właściwości zdefiniowane przez aplikację, które nie są ujawniane jako właściwości zdefiniowane przez użytkownika produktu JMS . Właściwości, które nie spełniają wymagań produktu JMS , mogą zostać umieszczone w tym folderze.

Istnieje możliwość utworzenia właściwości w folderze mq_usr . Właściwości utworzone w produkcie mq_usr są podobne do właściwości tworzonych w nowych folderach z atrybutem content='properties' .

sib

Produkt sib zawiera właściwości komunikatu systemowego magistrali integracji usług (WAS/SIB) produktu WebSphere Application Server . Właściwości produktu sib nie są ujawniane jako właściwości produktu JMS dla aplikacji IBM MQ JMS , ponieważ nie są one obsługiwane przez obsługiwane typy. Na przykład niektóre właściwości produktu sib nie mogą być prezentowane jako właściwości produktu JMS , ponieważ są to tablice bajtów. Niektóre właściwości produktu sib są narażone na działanie aplikacji WAS/SIB jako właściwości produktu JMS_IBM_* . Te właściwości obejmują właściwości ścieżek routingu do przodu i do tyłu.

Nie należy dodawać własnych właściwości w folderze sib.

sib_context

Produkt sib_context zawiera właściwości komunikatów systemowych WAS/SIB, które nie są ujawnione dla aplikacji użytkownika WAS/SIB lub jako właściwości produktu JMS . sib_context zawiera zabezpieczenia i właściwości transakcyjne, które są używane dla usług Web Service.

Nie należy dodawać własnych właściwości w folderze sib_context.

sib_usr

Produkt sib_usr zawiera właściwości komunikatu użytkownika WAS/SIB, które nie są ujawniane jako właściwości użytkownika produktu JMS , ponieważ nie są obsługiwane. Produkt sib_usr jest narażony na działanie aplikacji WAS/SIB w interfejsie produktu SIMessage . Patrz [Tworzenie integracji usług](#).

Typem właściwości sib_usr musi być bin.hex, a jej wartość musi być w poprawnym formacie. Jeśli aplikacja IBM MQ zapisze element typu bin.hex do folderu w niewłaściwym formacie, aplikacja otrzymuje IOException. Jeśli typem danych właściwości jest inny niż bin.hex , aplikacja otrzymuje ClassCastException.

Nie należy podejmować próby udostępnienia właściwości użytkownika produktu JMS WAS/SIB przy użyciu tego folderu. Zamiast tego należy użyć folderu usr .

Istnieje możliwość utworzenia właściwości w folderze sib_usr .

usr

Folder usr zawiera zdefiniowane przez aplikację właściwości JMS powiązane z komunikatem. Folder usr jest obecny tylko wtedy, gdy w aplikacji ustawiono właściwość definiowaną przez aplikację.

usr jest domyślnym folderem właściwości. Jeśli właściwość jest ustawiona bez nazwy folderu, jest ona umieszczana w folderze usr .

Tabela 73. usr - nazwa właściwości, synonim, typ danych i folder.

Wartości właściwości usług Web Service są opisane w sekcji [Ustawienia SOAP produktu MQRFH2](#) .

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	usr.contentType	string	<usr><contentType>text/xml; charset=utf-8</contentType></usr>

Tabela 73. *usr* - nazwa właściwości, synonim, typ danych i folder.

Wartości właściwości usług Web Service są opisane w sekcji [Ustawienia SOAP produktu MQRFH2](#) .
(kontynuacja)

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	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>

Istnieje możliwość utworzenia właściwości w folderze *usr* .

Posegmentowany komunikat, który zawiera MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED , nie może zawierać MQRFH2 o zdefiniowanej nazwie folderu właściwości. MQPUT nie powiodło się z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Niezgrupowana nazwa folderu właściwości

ibm

ibm zawiera właściwości, które są używane tylko przez produkt IBM MQ.

Tabela 74. *ibm* - nazwa właściwości, synonim, typ danych i folder

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

Nie należy dodawać własnych właściwości w folderze *ibm*.

mq

mq zawiera właściwości, które są używane tylko przez produkt IBM MQ.

Do właściwości w folderze *mq* mają zastosowanie następujące ograniczenia:

- Tylko właściwości w pierwszym znaczącym folderze *mq* w komunikacie są zachowane przez produkt MQ; właściwości w dowolnym innym folderze *mq* w komunikacie są ignorowane.
- W folderze dozwolone są tylko znaki UTF-8 jednobajtowe. Wielobajtowy znak w folderze, może spowodować niepowodzenie analizowania, a komunikat zostanie odrzucony.
- W folderze nie należy używać łańcuchów zmiany znaczenia. Łańcuch zmiany znaczenia jest traktowany jako rzeczywista wartość elementu.
- Tylko znak Unicode U+0020 jest traktowany jako biały znak w folderze. Wszystkie inne znaki są traktowane jako znaczące i mogą spowodować niepowodzenie analizowania folderu, a komunikat do odrzucenia.

Jeśli analizowanie folderu mq nie powiedzie się lub jeśli folder nie będzie obserwował tych ograniczeń, komunikat zostanie odrzucony z kodem przyczyny 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR.

Nie należy dodawać własnych właściwości w folderze mq.

mqema

mqema zawiera właściwości, które są używane tylko przez produkt WebSphere Application Server. Folder został zastąpiony przez produkt mqext.

Nie należy dodawać własnych właściwości w folderze mqema.

mqext

Folder mqext zawiera następujące typy właściwości:

- Właściwości, które są używane tylko przez produkt WebSphere Application Server.
- Właściwości związane z opóźnionym dostarczaniem komunikatów.

Folder jest obecny, jeśli w przypadku aplikacji ustawiono co najmniej jedną właściwość zdefiniowaną przez IBM albo używane jest opóźnienie dostawy.

Tabela 75. Synonim, typ danych, folder i nazwa właściwości mqext			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>armCorrelator</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>wrmCorrelator</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>DeliveryTime</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>DeliveryTime</Dly></mqext>

Nie należy dodawać własnych właściwości w folderze mqext.

mqps

Folder mqps zawiera właściwości, które są używane tylko w operacjach publikowania/subskrybowania programu IBM MQ. Folder jest obecny tylko wtedy, gdy w przypadku aplikacji ustawiono co najmniej jedną zintegrowaną właściwość publikowania/subskrybowania.

Tabela 76. Synonim, typ danych, folder i nazwa właściwości mqps			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
MQTopicString	mqps.Top	string	<mqps><Top>topicString</Top></mqps>
MQSubUserData	mqps.Sud	string	<mqps><Sud>subscriberUserData</Sud></mqps>
MQIsRetained	mqps.Ret	boolean	<mqps><Ret>isRetained</Ret></mqps>
MQPubOptions	mqps.Pub	i8	<mqps><Pub>publicationOptions</Pub></mqps>
MQPubLevel	mqps.Pbl	i8	<mqps><Pbl>publicationLevel</Pbl></mqps>
MQPubTime	mqps.Pts	string	<mqps><Pts>publicationTime</Pts></mqps>

Tabela 76. Synonim, typ danych, folder i nazwa właściwości mqps (kontynuacja)			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
MQPubSeqNum	mqpse.Seq	i8	<mqps><Seq>publicationSequenceNumber</Seq></mqps>
MQPubStrIntData	mqpse.Sid	string	<mqps><Sid>publicationData</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>messageFormat</Pfmt></mqps>

Nie należy dodawać własnych właściwości w folderze mqps.

mq_svc

Produkt mq_svc zawiera właściwości używane przez pakiet SupportPac MA93.

Nie należy dodawać własnych właściwości w folderze mq_svc.

mqtt

Produkt mqtt zawiera właściwości używane przez produkt IBM MQ Telemetry .

Tabela 77. mqtt - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	mqtt.clientId	string	<mqtt><clientId>topicString</clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos>qualityOfService</qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid>messageIdentifier</msgid></mqtt>

Nie należy dodawać własnych właściwości w folderze mqtt.

Segmentowany komunikat o nazwie MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED nie może zawierać MQRFH2 z niezgrupowaną nazwą folderu właściwości. MQPUT nie powiodło się z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Pary nazwa/wartość

W diagramie składniowym "pary nazwa/wartość" opisują treść folderu zwykłego. Zwykły folder zawiera grupy i elementy. Element jest parą nazwy/wartość. Grupa zawiera elementy i inne grupy.

W przypadku drzew elementy są węzłami liści, a grupy są węzłami wewnętrznymi. Węzeł wewnętrzny i folder, który jest węzłem głównym, mogą zawierać kombinację węzłów wewnętrznych i liści. Węzeł nie może być jednocześnie węzłem wewnętrznym i liściowym; patrz [Rysunek 2 na stronie 536](#).

Właściwości

W diagramie składniowym "Właściwości" opisuje treść folderu właściwości. Folder właściwości zawiera grupy i właściwości. Właściwość jest parą nazwy/wartość z opcjonalnym atrybutem typu danych. Grupa zawiera właściwości i inne grupy.

W przypadku drzew właściwości są węzłami liści, a grupy są węzłami wewnętrznymi. Węzeł wewnętrzny i folder właściwości, który jest węzłem głównym, mogą zawierać mieszaninę węzłów wewnętrznych i liści. Węzeł nie może być jednocześnie węzłem wewnętrznym i liściowym; patrz [Rysunek 2 na stronie 536](#).

Właściwość

Właściwość komunikatu to para nazwa/wartość w folderze właściwości. Opcjonalnie może on zawierać atrybut typu danych i atrybut właściwości; na przykład można wyświetlić następujący kod. Jeśli atrybut typu danych zostanie pominięty, typem właściwości jest `string`.

```
<pf><p1 dt='i8' > value </p1></pf>
```

Nazwa właściwości komunikatu jest pełną nazwą ścieżki, przy czym składnia XML jest następująca: `<>`, zastępowana przez kropki. Na przykład `myPropertyFolder1.myGroup1.myGroup2.myProperty1` jest odwzorowywany na łańcuch `NameValueData` w następujący sposób. Łańcuch jest sformatowany w celu łatwiejszego odczytu.

```
<myPropertyFolder1>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
    </myGroup2>
  </myGroup1>
</myPropertyFolder1>
```

Folder właściwości może zawierać wiele właściwości. Na przykład właściwości w produkcie [Rysunek 7 na stronie 543](#) są odwzorowywane na folder właściwości w produkcie [Rysunek 8 na stronie 543](#).

```
myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Rysunek 7. Wiele właściwości o tej samej nazwie użytkownika root

```
<myPropertyFolder1>
  <myProperty4>value</myProperty4>
  <myGroup1>
    <myGroup2>
      <myProperty1>value</myProperty1>
      <myProperty2>value</myProperty2>
    </myGroup2>
    <myProperty3>value</myProperty3>
  </myGroup1>
</myPropertyFolder1>
```

Rysunek 8. Odwzorowanie nazwy wielu właściwości

Nazwa

Nazwa musi zaczynać się od litery *Letter* lub *Underscore*. Nie może zawierać *Colon*, nie może kończyć się w *okresie* i może zawierać tylko litery *Letters*, *Numerals*, *Underscores*, *Hyphens* i *Dots*. Poprawne znaki są zdefiniowane w specyfikacji XML W3C i składają się głównie z kategorii Unicode `L1`, `Lu`, `Lo`, `Lt`, `N1`, `Mc`, `Mn`, `Lm`, i `Nd`; patrz [Kategorie znaków Unicode](#).

Pełna ścieżka do pary właściwości lub nazwy/wartości nie może złamać reguły opisanej w [“Niepoprawna nazwa ścieżki” na stronie 546](#). Ścieżki są ograniczone do 4095 bajtów, nie mogą zawierać znaków zgodności z kodami Unicode i nie mogą zaczynać się od łańcucha XML.

Nazwa grupy

Nazwa grupy ma taką samą składnię, jak nazwa. Nazwy grup są opcjonalne. Właściwości i pary nazwa/wartość mogą być umieszczane w katalogu głównym folderu. Użyj grup, jeśli pomagają w organizowaniu par właściwościom i nazwa/wartość.

Nazwa elementu

Nazwa elementu ma taką samą składnię, jak nazwa.

Wartość elementu

Wartość elementu obejmuje całą białą przestrzeń między znacznikiem `< Element name >` i `< / Element name >`. Nie należy używać dwóch znaków `<` i `&` w wartości. Zastąp następnie `<` i `&`;

Atrybuty elementu Property

Atrybuty właściwości odwzorowują pola deskryptora właściwości: odwzorowania są następujące:

Obsługa

sa	MQPD_SUPPORT_OPTIONAL
sr	MQPD_SUPPORT_REQUIRED
sx	MQPD_SUPPORT_REQUIRED_IF_LOCAL

Kontekst

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

Opcji **all** nie należy używać w połączeniu z innymi opcjami.

default	MQPD_COPY_DEFAULT

Opcji **default** nie należy używać w połączeniu z innymi opcjami. **default** jest taki sam jak **forward** + **report** + **publish**

none	MQPD_COPY_NONE

Opcji **none** nie należy używać w połączeniu z innymi opcjami.

Atrybuty właściwości **Support** mają zastosowanie tylko do właściwości w folderze **mq**.

Atrybuty właściwości **Kontekst** i **CopyOptions** mają zastosowanie do wszystkich folderów właściwości.

Typy danych

Typy danych produktu MQRFH2 są odwzorowywać na typy właściwości komunikatów w następujący sposób:

Tabela 78. Odwzorowania typów danych	
MQRFH2 Typ danych	Typ właściwości komunikatu
bin.hex	MQBYTE[]
boolean	MQBOOL
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Przyjmuje się, że każdy element bez typu danych ma typ string.

Wartość NULL jest wskazywana przez atrybut elementu `xsi:nil='true'`. Nie należy używać atrybutu `xsi:nil='false'` dla wartości innych niż NULL. Na przykład następująca właściwość ma wartość NULL:

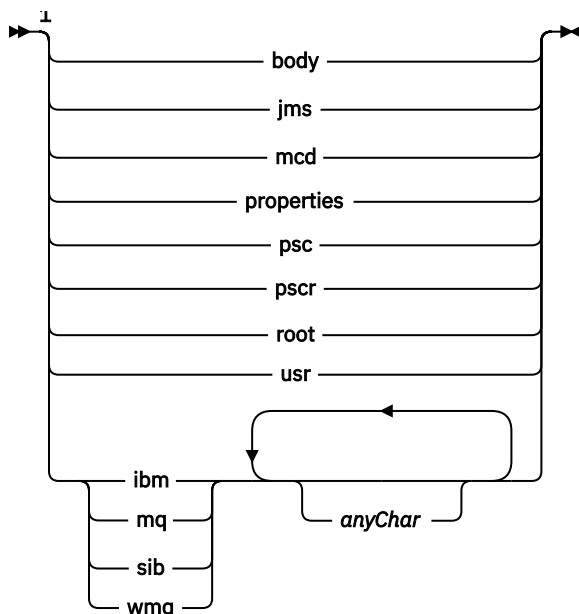
```
<NullProperty  
xsi:nil='true'></NullProperty>
```

Właściwość typu byte lub łańcuch znaków może mieć pustą wartość. Pusta wartość jest reprezentowana przez element MQRFH2 o wartości elementu o zerowej długości. Na przykład następująca właściwość ma pustą wartość:

```
<EmptyProperty></EmptyProperty>
```

Zastrzeżony folder lub nazwa folderu właściwości

Ogranicz nazwę folderu lub folderu właściwości, aby nie rozpoczynać się od żadnego z następujących łańcuchów. Przedrostki są zarezerwowane dla nazw folderów lub właściwości utworzonych przez produkt IBM.



Uwagi:

¹ Zastrzeżony folder lub nazwa właściwości zawiera dowolną mieszaninę małych i wielkich liter.

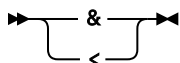
Niepoprawna nazwa ścieżki

Ogranicz pełną ścieżkę do pary nazwa/wartość lub właściwość, aby nie zawierała żadnego z następujących łańcuchów.



Nieprawidłowe znaki

Zawsze należy używać sekwencji o zmienionym znaczeniu & i < zamiast literałów "&" i "<".

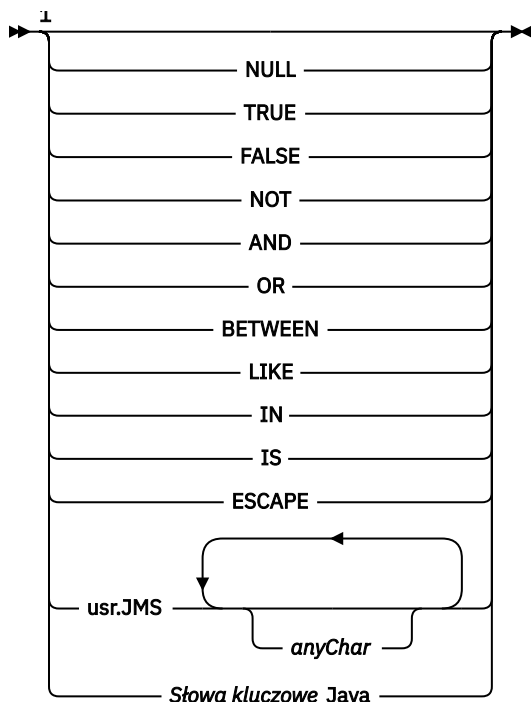


Zdefiniowane nazwy właściwości

Zdefiniowane nazwy właściwości to nazwy właściwości, które są definiowane przez produkt IBM MQ lub inne produkty i używane przez produkt IBM MQ i aplikacje użytkownika. Zdefiniowane właściwości istnieją tylko w zdefiniowanych folderach właściwości. Zdefiniowane nazwy właściwości są opisane w opisie folderów właściwości. Patrz sekcja [Foldery właściwości](#).

Niepoprawna nazwa właściwości

Nie należy konstruować nazw właściwości, które są zgodne z następującą regułą. Reguła ma zastosowanie do pełnej ścieżki właściwości, która określa nazwę właściwości, a nie tylko do nazwy elementu właściwości.



Uwagi:

¹ Niepoprawna nazwa właściwości może zawierać dowolną kombinację wielkich i małych liter.

Niepoprawne atrybuty

Foldery właściwości i właściwości mogą zawierać tylko obsługiwane [“Atrybuty elementu Property”](#) na stronie 544 i [“Typy danych”](#) na stronie 545.

Wszystkie nieobsługiwane atrybuty podobne do XML, na przykład nazwy z wartościami łańcuchowymi ujętymi w cudzysłowy, które znajdują się w folderach właściwości lub właściwościach, mogą zostać usunięte.

Atrybuty podobne do XML zawarte w folderach innych niż właściwości lub elementy niezwiązane z właściwością, które pozostają w nagłówkach MQRFH2 .

NameValue(długość nazwy) (MQLONG)

Długość odpowiedniego pola *NameValueData*

Określa długość danych w bajtach w polu *NameValueData* . *NameValueLength* musi być wielokrotnością czterech.

Uwaga: Pola *NameValueLength* i *NameValueData* są opcjonalne, ale jeśli występują, muszą one występować jako para i być sąsiadującymi. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ pola te są opcjonalne, są one pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQRFH_STRUC_ID

Identyfikator regułu i struktury nagłówka formatowania.

W przypadku języka programowania C jest również zdefiniowana stała `MQRFH_STRUC_ID_ARRAY`. Ma ona taką samą wartość jak `MQRFH_STRUC_ID`, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to `MQRFH_STRUC_ID`.

StrucLength (MQLONG)

Jest to długość w bajtach struktury `MQRFH2`, w tym pola *NameValueLength* i *NameValueData* na końcu struktury. Ważne jest, aby na końcu struktury było wiele par pól *NameValueLength* i *NameValueData*, w kolejności:

```
length1, data1, length2, data2, ...
```

StrucLength nie zawiera żadnych danych użytkownika, które mogą być zgodne z ostatnim polem *NameValueData* na końcu struktury.

Aby uniknąć problemów z przekształceniem danych użytkownika w niektórych środowiskach, produkt *StrucLength* musi być wielokrotnością liczby czterech.

Następująca stała daje długość stałej części struktury, to znaczy długości wykluczając pola *NameValueLength* i *NameValueData*:

MQRFH_STRUC_LENGTH_FIXED_2

Długość stałej części struktury `MQRFH2`.

Początkowa wartość tego pola to `MQRFH_STRUC_LENGTH_FIXED_2`.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQRFH_VERSION_2

Reguły Version-2 i struktura nagłówka formatowania.

Początkowa wartość tego pola to `MQRFH_VERSION_2`.

Wartości początkowe i deklaracje języka dla MQRFH2

Tabela 79. Początkowe wartości pól w MQRFH2 dla MQRFH2		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	<code>MQRFH_STRUC_ID</code>	'RFH↵'
<i>Version</i>	<code>MQRFH_VERSION_2</code>	2
<i>StrucLength</i>	<code>MQRFH_STRUC_LENGTH_FIXED_2</code>	36
<i>Encoding</i>	<code>MQENC_NATIVE</code>	Zależy od środowiska
<i>CodedCharSetId</i>	<code>MQCCSI_INHERIT</code>	-2
<i>Format</i>	<code>MQFMT_NONE</code>	Puste
<i>Flags</i>	<code>MQRFH_NONE</code>	0
<i>NameValueCCSID</i>	Brak	1208

Tabela 79. Początkowe wartości pól w MQRFH2 dla MQRFH2 (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: - Symbol – reprezentuje pojedynczy pusty znak. - W języku programowania C: zmienna makraMQRFH2_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};		

Deklaracja C

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;          /* Structure version number */
    MQLONG    StrucLength;       /* Total length of MQRFH2 including all
                                NameValueLength and NameValueData
                                fields */
    MQLONG    Encoding;         /* Numeric encoding of data that follows
                                last NameValueData field */
    MQLONG    CodedCharSetId;    /* Character set identifier of data that
                                follows last NameValueData field */
    MQCHAR8    Format;          /* Format name of data that follows last
                                NameValueData field */
    MQLONG    Flags;            /* Flags */
    MQLONG    NameValueCCSID;    /* Character set identifier of
                                NameValueData */
};
```

Deklaracja języka COBOL

```
** MQRFH2 structure
10 MQRFH2.
** Structure identifier
15 MQRFH2-STRUCID PIC X(4).
** Structure version number
15 MQRFH2-VERSION PIC S9(9) BINARY.
** Total length of MQRFH2 including all NAMEVALUELENGTH and
** NAMEVALUEDATA fields
15 MQRFH2-STRUCLNGTH PIC S9(9) BINARY.
** Numeric encoding of data that follows last NAMEVALUEDATA field
15 MQRFH2-ENCODING PIC S9(9) BINARY.
** Character set identifier of data that follows last NAMEVALUEDATA
** field
15 MQRFH2-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of data that follows last NAMEVALUEDATA field
15 MQRFH2-FORMAT PIC X(8).
** Flags
15 MQRFH2-FLAGS PIC S9(9) BINARY.
** Character set identifier of NAMEVALUEDATA
15 MQRFH2-NAMEVALUECCSID PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQRFH2 based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 StrucLength fixed bin(31), /* Total length of MQRFH2 including
                                all NameValueLength and
                                NameValueData fields */
3 Encoding fixed bin(31), /* Numeric encoding of data that
                                follows last NameValueData field */
3 CodedCharSetId fixed bin(31), /* Character set identifier of data
```

```

that follows last NameValueData
field */
3 Format          char(8),      /* Format name of data that follows
last NameValueData field */
3 Flags          fixed bin(31), /* Flags */
3 NameValueCCSID fixed bin(31); /* Character set identifier of
NameValueData */

```

Deklaracja High Level Assembler

```

MQRFH          DSECT
MQRFH_STRUCID   DS    CL4      Structure identifier
MQRFH_VERSION   DS    F        Structure version number
MQRFH_STRUCLNGTH 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)

```

Wizualna deklaracja podstawowa

```

Type MQRFH2
  StrucId       As String*4 'Structure identifier'
  Version       As Long      'Structure version number'
  StrucLength   As Long      'Total length of MQRFH2 including all'
                               'NameValueLength and NameValueData fields'
  Encoding      As Long      'Numeric encoding of data that follows'
                               'last NameValueData field'
  CodedCharSetId As Long      'Character set identifier of data that'
                               'follows last NameValueData field'
  Format         As String*8 'Format name of data that follows last'
                               'NameValueData field'
  Flags         As Long      'Flags'
  NameValueCCSID As Long      'Character set identifier of NameValueData'
End Type

```

MQRMH-nagłówek komunikatu odwołania

W poniższej tabeli podsumowano pola w strukturze.

Tabela 80. Pola w MQRMH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Całkowita długość MQRMH, w tym łańcuchy na końcu pól stałych, ale nie dane masowe	StrucLength
<i>Encoding</i>	Kodowanie numeryczne danych masowych	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków danych masowych	CodedCharSetId
<i>Format</i>	Nazwa formatu danych masowych	Formatowanie
<i>Flags</i>	Flagi komunikatów odniesienia	Flagi

Tabela 80. Pola w MQRMH (kontynuacja)

Pole	Opis	Temat
<i>ObjectType</i>	Typ obiektu	ObjectType
<i>ObjectInstanceId</i>	Identyfikator instancji obiektu	IdentyfikatorObjectInstancje
<i>SrcEnvLength</i>	Długość danych środowiska źródłowego	SrcEnvDługość
<i>SrcEnvOffset</i>	Przesunięcie danych środowiska źródłowego	PrzesunięcieSrcEnv
<i>SrcNameLength</i>	Długość nazwy obiektu źródłowego	SrcNameDługość
<i>SrcNameOffset</i>	Przesunięcie nazwy obiektu źródłowego	SrcNamePrzesunięcie
<i>DestEnvLength</i>	Długość danych środowiska docelowego	DestEnvDługość
<i>DestEnvOffset</i>	Przesunięcie danych środowiska docelowego	DestEnvPrzesunięcie
<i>DestNameLength</i>	Długość nazwy obiektu docelowego	DestNameDługość
<i>DestNameOffset</i>	Przesunięcie nazwy obiektu docelowego	DestNamePrzesunięcie
<i>DataLogicalLength</i>	Długość danych masowych	DataLogicalDługość
<i>DataLogicalOffset</i>	Niskie przesunięcie danych masowych	DataLogicalPrzesunięcie
<i>DataLogicalOffset2</i>	Wysokie przesunięcie danych masowych	DataLogicalOffset2

Przegląd produktu MQRMH

Dostępność: klienci AIX, HP-UX, IBM i, Solaris, Linux, Windowsi IBM MQ połączone z tymi systemami.

Cel: Struktura MQRMH definiuje format nagłówka komunikatu odniesienia. Ten nagłówek jest używany z wyjściami kanału komunikatów napisanych przez użytkownika w celu wysyłania bardzo dużych ilości danych (nazywanych *danymi masowymi*). z jednego menedżera kolejek do innego. Różnica w porównaniu z normalnym przesyłaniem komunikatów polega na tym, że dane masowe nie są zapisywane w kolejce. Zamiast tego w kolejce przechowywane są tylko *odwołanie* do danych masowych. Zmniejsza to możliwość wyczerpania zasobów MQ przez niewielką liczbę bardzo dużych komunikatów.

Nazwa formatu: MQFMT_REF_MSG_HEADER.

Zestaw znaków i kodowanie: Dane znakowe w MQRMH oraz łańcuchy adresowane przez pola przesunięcia muszą znajdować się w zestawie znaków lokalnego menedżera kolejek. Ten atrybut jest nadawany przez atrybut menedżera kolejek produktu *CodedCharSetId*. Dane liczbowe w MQRMH muszą znajdować się w rodzimym kodowaniu komputera; jest to nadawane przez wartość MQENC_NATIVE dla języka programowania C.

Ustaw zestaw znaków i kodowanie wartości MQRMH w polach *CodedCharSetId* i *Encoding* w:

- MQMD (jeśli struktura MQRMH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQRMH (wszystkie inne obserwacje).

Użycie: aplikacja umieszcza komunikat składający się z wywołania MQRMH, ale pomija dane masowe. Gdy agent kanału komunikatów (MCA) odczytuje komunikat z kolejki transmisji, wywoływany przez użytkownika wyjście komunikatów jest wywoływane w celu przetworzenia nagłówka komunikatu odniesienia. Wyjście może dopisać do komunikatu referencyjnego dane masowe identyfikowane przez strukturę MQRMH, zanim agent MCA wyśle komunikat przez kanał do następnego menedżera kolejek.

Po zakończeniu odbierania musi istnieć wyjście komunikatu, które oczekuje na komunikaty odniesienia. Po odebraniu komunikatu referencyjnego wyjście musi utworzyć obiekt na podstawie danych masowych, które są następujące po komunikacie MQRMH w komunikacie, a następnie przekazać komunikat odwołania bez danych masowych. Komunikat referencyjny może zostać później pobrany przez aplikację odczytując komunikat odniesienia (bez danych masowych) z kolejki.

Zwykle struktura MQRMH jest wszystkim, co znajduje się w komunikacie. Jeśli jednak komunikat znajduje się w kolejce transmisji, to jeden lub więcej dodatkowych nagłówków poprzedza strukturę MQRMH.

Komunikat odniesienia może również zostać wysłany do listy dystrybucyjnej. W tym przypadku struktura MQDH i powiązane z nią rekordy poprzedzają strukturę MQRMH, gdy komunikat znajduje się w kolejce transmisji.

Uwaga: Nie wysyłaj komunikatu referencyjnego jako posegmentowanego komunikatu, ponieważ wyjście komunikatu nie może przetworzyć tego komunikatu poprawnie.

Konwersja danych: W celu konwersji danych struktura MQRMH jest przekształcana w konwersję danych środowiska źródłowego, nazwy obiektu źródłowego, danych środowiska docelowego i nazwy obiektu docelowego. Wszystkie pozostałe bajty w bajtach *StrucLength* początku struktury są odrzucane lub mają niezdefiniowane wartości po konwersji danych. Dane masowe są przekształcane pod warunkiem, że spełnione są wszystkie poniższe warunki:

- Dane masowe są obecne w komunikacie, gdy wykonywana jest konwersja danych.
- Pole *Format* w tabeli MQRMH ma wartość inną niż MQFMT_NONE.
- W przypadku wyjścia konwersji danych napisanych przez użytkownika istnieje określona nazwa formatu.

Należy jednak pamiętać, że zwykle dane masowe *nie* są obecne w komunikacie, gdy komunikat znajduje się w kolejce, a w rezultacie dane masowe są przekształcane za pomocą opcji MQGMO_CONVERT.

Pola dla MQRMH

Struktura MQRMH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CodedCharSetId (MQLONG)

Określa identyfikator zestawu znaków danych masowych; nie ma zastosowania do danych znakowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie należy używać wartości MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcie MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz klienty IBM MQ połączone z tymi systemami.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Długość DataLogical(MQLONG)

Pole *DataLogicalLength* określa długość danych masowych, do których odwołuje się struktura MQRMH.

Jeśli dane masowe są rzeczywiście obecne w komunikacie, dane zaczynają się od przesunięcia *StrucLength* bajtów od początku struktury MQRMH. Długość całego komunikatu pomniejszona o *StrucLength* określa długość danych masowych.

Jeśli dane są obecne w komunikacie, *DataLogicalLength* określa ilość danych, które są istotne. Normalny przypadek dotyczy wartości *DataLogicalLength*, która ma taką samą wartość jak długość danych znajdujących się w komunikacie.

Jeśli struktura MQRMH reprezentuje pozostałe dane w obiekcie (począwszy od określonego przesunięcia logicznego), można użyć wartości zero dla *DataLogicalLength*, pod warunkiem, że dane masowe nie są rzeczywiście obecne w komunikacie.

Jeśli nie ma żadnych danych, koniec komunikatu MQRMH jest zbieżny z końcem komunikatu.

Wartością początkową tego pola jest 0.

Przesunięcie DataLogical(MQLONG)

To pole określa niską wartość przesunięcia danych masowych od początku obiektu, którego część stanowi część danych masowych. Przesunięcie danych masowych od początku obiektu jest nazywane *przesuniętym logicznym*. Jest to *nie* przesunięcie fizyczne danych masowych od początku struktury MQRMH; przesunięcie to jest nadawane przez produkt *StrucLength*.

Aby umożliwić wysyłanie dużych obiektów za pomocą komunikatów referencyjnych, przesunięcie logiczne jest podzielone na dwa pola, a rzeczywiste przesunięcie logiczne jest nadawane przez sumę tych dwóch pól:

- *DataLogicalOffset* reprezentuje pozostałą część otrzymaną, gdy przesunięcie logiczne dzieli się na 1 000 000 000. Jest to zatem wartość z zakresu od 0 do 999 999 999.
- *DataLogicalOffset2* reprezentuje wynik uzyskany, gdy przesunięcie logiczne dzieli się na 1 000 000 000. Jest to zatem liczba pełnych wielokrotności 1 000 000 000 istniejących w logice offsetowej. Liczba wielokrotności mieści się w zakresie od 0 do 999 999 999.

Wartością początkową tego pola jest 0.

DataLogicalOffset2 (MQLONG)

To pole określa wysokie przesunięcie danych masowych od początku obiektu, którego część stanowi część danych masowych. Jest to wartość z zakresu od 0 do 999 999 999. Szczegółowe informacje można znaleźć w sekcji *DataLogicalOffset*.

Wartością początkową tego pola jest 0.

Długość DestEnv(MQLONG)

Jest to długość danych środowiska docelowego. Jeśli to pole ma wartość zero, dane środowiska docelowego nie są dostępne, a parametr *DestEnvOffset* jest ignorowany.

Przesunięcie DestEnv(MQLONG)

To pole służy do określania przesunięcia danych środowiska docelowego z początku struktury MQRMH. Dane środowiska docelowego mogą być określone przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Na przykład, w systemie Windows dane środowiska docelowego mogą być ścieżką do katalogu obiektu, w którym mają być przechowywane dane masowe. Jeśli jednak twórca nie zna danych środowiska docelowego, jest on odpowiedzialny za wyjście komunikatów dostarczone przez użytkownika w celu określenia wszelkich potrzebnych informacji o środowisku.

Długość danych środowiska docelowego jest podawana przez produkt *DestEnvLength* ; Jeśli ta długość wynosi zero, nie ma danych środowiska docelowego, a program *DestEnvOffset* jest ignorowany. Jeśli istnieje, dane środowiska docelowego muszą znajdować się całkowicie w bajtach *StrucLength* od początku struktury.

Aplikacje nie mogą zakładać, że dane środowiska docelowego są ciągłe w przypadku danych adresowanych przez pola *SrcEnvOffset*, *SrcNameOffset* i *DestNameOffset* .

Wartością początkową tego pola jest 0.

DestName(MQLONG)

Długość nazwy obiektu docelowego. Jeśli to pole ma wartość zero, nie ma nazwy obiektu docelowego, a parametr *DestNameOffset* jest ignorowany.

DestNamePrzesunięcie (MQLONG)

To pole określa przesunięcie nazwy obiektu docelowego od początku struktury MQRMH. Nazwa obiektu docelowego może być określona przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu docelowego, jest on odpowiedzialny za wyjście z komunikatu

dostarczonego przez użytkownika w celu zidentyfikowania obiektu, który ma zostać utworzony lub zmodyfikowany.

Długość nazwy obiektu docelowego jest podawana przez produkt *DestNameLength* ; Jeśli ta długość wynosi zero, nie istnieje nazwa obiektu docelowego, a parametr *DestNameOffset* jest ignorowany. Jeśli ta wartość jest obecna, nazwa obiektu docelowego musi znajdować się całkowicie w bajtach *StrucLength* od początku struktury.

Aplikacje nie mogą zakładać, że nazwa obiektu docelowego jest ciągła z dowolnym z danych adresowanych przez pola *SrcEnvOffset*, *SrcNameOffset* i *DestEnvOffset* .

Wartością początkową tego pola jest 0.

Kodowanie (MQLONG)

Określa kodowanie numeryczne danych masowych; nie ma zastosowania do danych liczbowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

Flagi (MQLONG)

Są to flagi komunikatów odniesienia. Zdefiniowane są następujące opcje:

MQRMHF_LAST

Ta opcja wskazuje, że komunikat odniesienia reprezentuje ostatnią część obiektu, do którego istnieje odwołanie.

MQRMHF_NOT_LAST

Komunikat odniesienia nie zawiera ani nie reprezentuje ostatniej części obiektu. Dokumentacja programu pomocy MQRMHF_NOT_LAST. Ta opcja nie jest przeznaczona do użycia z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest MQRMHF_NOT_LAST.

Format (MQCHAR8)

Określa nazwę formatu danych masowych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

Identyfikator ObjectInstance(MQBYTE24)

To pole służy do identyfikowania konkretnej instancji obiektu. Jeśli nie jest to potrzebne, należy ustawić wartość na następujące wartości:

MQOII_NONE

Nie określono identyfikatora instancji obiektu. Wartość jest binarna zero dla długości pola.

Dla języka programowania C definiowana jest także stała MQOII_NONE_ARRAY; ma ona taką samą wartość jak MQOII_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_OBJECT_INSTANCE_ID_LENGTH. Wartością początkową tego pola jest MQOII_NONE.

ObjectType (MQCHAR8)

Jest to nazwa, której program obsługi wyjścia może używać do rozpoznawania typów komunikatów referencyjnych obsługiwanych przez ten program. Nazwa musi być zgodna z tymi samymi regułami, co w polu *Format* , patrz ["Format \(MQCHAR8\)" na stronie 554](#).

Początkowa wartość tego pola wynosi 8 znaków odstępu.

Długość SrcEnv(MQLONG)

Długość danych środowiska źródłowego. Jeśli to pole ma wartość zero, nie ma danych środowiska źródłowego, a program *SrcEnvOffset* jest ignorowany.

Wartością początkową tego pola jest 0.

SrcEnvPrzesunięcie (MQLONG)

To pole określa przesunięcie źródła danych środowiska źródłowego od początku struktury MQRMH. Dane środowiska źródłowego mogą być określone przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcom. Na przykład w systemie Windows dane środowiska źródłowego mogą być ścieżką do katalogu obiektu zawierającego dane masowe. Jeśli jednak twórca nie zna danych środowiska źródłowego, program zewnętrzny dostarczony przez użytkownika musi określić wymagane informacje o środowisku.

Długość danych środowiska źródłowego jest podawana przez produkt *SrcEnvLength* ; Jeśli ta długość wynosi zero, nie ma danych środowiska źródłowego, a program *SrcEnvOffset* jest ignorowany. Jeśli jest to obecne, dane środowiska źródłowego muszą znajdować się całkowicie w bajtach *StrucLength* od początku struktury.

Aplikacje nie mogą zakładać, że dane środowiska zaczynają się od razu po ostatnim stałym polu w strukturze lub że są one przylegające do dowolnych danych adresowanych przez pola *SrcNameOffset*, *DestEnvOffset* i *DestNameOffset* .

Wartością początkową tego pola jest 0.

SrcNameDługość (MQLONG)

Długość nazwy obiektu źródłowego. Jeśli to pole ma wartość zero, nie istnieje żadna nazwa obiektu źródłowego, a parametr *SrcNameOffset* jest ignorowany.

Wartością początkową tego pola jest 0.

SrcNamePrzesunięcie (MQLONG)

To pole określa przesunięcie nazwy obiektu źródłowego od początku struktury MQRMH. Nazwa obiektu źródłowego może zostać określona przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcom. Jeśli jednak twórca nie zna nazwy obiektu źródłowego, program zewnętrzny dostarczony przez użytkownika musi zidentyfikować obiekt, do którego ma być uzyskany dostęp.

Długość nazwy obiektu źródłowego jest podawana przez produkt *SrcNameLength* ; Jeśli ta długość wynosi zero, nie istnieje żadna nazwa obiektu źródłowego, a parametr *SrcNameOffset* jest ignorowany. Jeśli ta opcja jest obecna, nazwa obiektu źródłowego musi znajdować się całkowicie w bajtach *StrucLength* od początku struktury.

Aplikacje nie mogą zakładać, że nazwa obiektu źródłowego jest ciągła z dowolnym z danych adresowanych przez pola *SrcEnvOffset*, *DestEnvOffset* i *DestNameOffset* .

Wartością początkową tego pola jest 0.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQRMH_STRUC_ID

Identyfikator struktury nagłówka komunikatu odniesienia.

W przypadku języka programowania C zdefiniowana jest również stała MQRMH_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQRMH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQRMH_STRUC_ID.

StrucLength (MQLONG)

Łączna długość wartości MQRMH, w tym łańcuchów na końcu pól stałych, ale nie danych masowych.

Początkowa wartość tego pola wynosi zero.

Wersja (MQLONG)

Numer wersji struktury. Wartość musi być następująca:

MQRMH_VERSION_1

Struktura nagłówka komunikatu odwołania Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQRMH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka komunikatu odwołania.

Początkowa wartość tego pola to MQRMH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQRMH

Tabela 81. Początkowe wartości pól w MQRMH dla MQRMH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQRMH_STRUC_ID	'RMH↵'
<i>Version</i>	MQRMH_VERSION_1	1
<i>StrucLength</i>	Brak	0
<i>Encoding</i>	MQENC_NATIVE	Zależy od środowiska
<i>CodedCharSetId</i>	MQCCSI_UNDEFINED	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Flags</i>	MQRMHF_NOT_LAST	0
<i>ObjectType</i>	Brak	Puste
<i>ObjectInstanceId</i>	MQOII_NONE	Wartości null
<i>SrcEnvLength</i>	Brak	0
<i>SrcEnvOffset</i>	Brak	0
<i>SrcNameLength</i>	Brak	0
<i>SrcNameOffset</i>	Brak	0
<i>DestEnvLength</i>	Brak	0
<i>DestEnvOffset</i>	Brak	0
<i>DestNameLength</i>	Brak	0
<i>DestNameOffset</i>	Brak	0
<i>DataLogicalLength</i>	Brak	0
<i>DataLogicalOffset</i>	Brak	0
<i>DataLogicalOffset2</i>	Brak	0

Tabela 81. Początkowe wartości pól w MQRMH dla MQRMH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Uwagi: - Symbol – reprezentuje pojedynczy pusty znak. - W języku programowania C: zmienna makraMQRMH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQRMH MyRMH = {MQRMH_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;       /* Total length of MQRMH, including
                                   strings at end of fixed fields, but
                                   not the bulk data */
    MQLONG     Encoding;         /* Numeric encoding of bulk data */
    MQLONG     CodedCharSetId;   /* Character set identifier of bulk
                                   data */
    MQCHAR8    Format;           /* Format name of bulk data */
    MQLONG     Flags;            /* Reference message flags */
    MQCHAR8    ObjectType;       /* Object type */
    MQBYTE24    ObjectInstanceId; /* Object instance identifier */
    MQLONG     SrcEnvLength;     /* Length of source environment data */
    MQLONG     SrcEnvOffset;     /* Offset of source environment data */
    MQLONG     SrcNameLength;    /* Length of source object name */
    MQLONG     SrcNameOffset;    /* Offset of source object name */
    MQLONG     DestEnvLength;    /* Length of destination environment
                                   data */
    MQLONG     DestEnvOffset;    /* Offset of destination environment
                                   data */
    MQLONG     DestNameLength;   /* Length of destination object name */
    MQLONG     DestNameOffset;   /* Offset of destination object name */
    MQLONG     DataLogicalLength; /* Length of bulk data */
    MQLONG     DataLogicalOffset; /* Low offset of bulk data */
    MQLONG     DataLogicalOffset2; /* High offset of bulk data */
};
```

Deklaracja języka COBOL

```
** MQRMH structure
10 MQRMH.
** Structure identifier
15 MQRMH-STRUCID PIC X(4).
** Structure version number
15 MQRMH-VERSION PIC S9(9) BINARY.
** Total length of MQRMH, including strings at end of fixed fields,
** but not the bulk data
15 MQRMH-STRUCLength PIC S9(9) BINARY.
** Numeric encoding of bulk data
15 MQRMH-ENCODING PIC S9(9) BINARY.
** Character set identifier of bulk data
15 MQRMH-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of bulk data
15 MQRMH-FORMAT PIC X(8).
** Reference message flags
15 MQRMH-FLAGS PIC S9(9) BINARY.
** Object type
15 MQRMH-OBJECTTYPE PIC X(8).
** Object instance identifier
15 MQRMH-OBJECTINSTANCEID PIC X(24).
** Length of source environment data
15 MQRMH-SRCENVLENGTH PIC S9(9) BINARY.
** Offset of source environment data
15 MQRMH-SRCENVOFFSET PIC S9(9) BINARY.
```

```

**      Length of source object name
15 MQRMH-SRCNAMELENGTH      PIC S9(9) BINARY.
**      Offset of source object name
15 MQRMH-SRCNAMEOFFSET      PIC S9(9) BINARY.
**      Length of destination environment data
15 MQRMH-DESTENVLENGTH      PIC S9(9) BINARY.
**      Offset of destination environment data
15 MQRMH-DESTENVOFFSET      PIC S9(9) BINARY.
**      Length of destination object name
15 MQRMH-DESTNAMELENGTH      PIC S9(9) BINARY.
**      Offset of destination object name
15 MQRMH-DESTNAMEOFFSET      PIC S9(9) BINARY.
**      Length of bulk data
15 MQRMH-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.

```

Deklaracja PL/I

```

dcl
1 MQRMH based,
3 StrucId          char(4),          /* Structure identifier */
3 Version          fixed bin(31),    /* Structure version number */
3 StrucLength      fixed bin(31),    /* Total length of MQRMH,
                                     including strings at end of
                                     fixed fields, but not the bulk
                                     data */
3 Encoding         fixed bin(31),    /* Numeric encoding of bulk
                                     data */
3 CodedCharSetId   fixed bin(31),    /* Character set identifier of
                                     bulk data */
3 Format            char(8),          /* Format name of bulk data */
3 Flags            fixed bin(31),    /* Reference message flags */
3 ObjectType       char(8),          /* Object type */
3 ObjectInstanceId char(24),         /* Object instance identifier */
3 SrcEnvLength     fixed bin(31),    /* Length of source environment
                                     data */
3 SrcEnvOffset     fixed bin(31),    /* Offset of source environment
                                     data */
3 SrcNameLength    fixed bin(31),    /* Length of source object name */
3 SrcNameOffset    fixed bin(31),    /* Offset of source object name */
3 DestEnvLength    fixed bin(31),    /* Length of destination
                                     environment data */
3 DestEnvOffset    fixed bin(31),    /* Offset of destination
                                     environment data */
3 DestNameLength   fixed bin(31),    /* Length of destination object
                                     name */
3 DestNameOffset   fixed bin(31),    /* Offset of destination object
                                     name */
3 DataLogicalLength fixed bin(31),    /* Length of bulk data */
3 DataLogicalOffset fixed bin(31),    /* Low offset of bulk data */
3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */

```

Deklaracja High Level Assembler

MQRMH	DSECT		
MQRMH_STRUCID	DS	CL4	Structure identifier
MQRMH_VERSION	DS	F	Structure version number
MQRMH_STRUCLength	DS	F	Total length of MQRMH, including
*			strings at end of fixed fields, but
*			not the bulk data
MQRMH_ENCODING	DS	F	Numeric encoding of bulk data
MQRMH_CODEDCHARSETID	DS	F	Character set identifier of bulk
*			data
MQRMH_FORMAT	DS	CL8	Format name of bulk data
MQRMH_FLAGS	DS	F	Reference message flags
MQRMH_OBJECTTYPE	DS	CL8	Object type
MQRMH_OBJECTINSTANCEID	DS	XL24	Object instance identifier
MQRMH_SRCENVLENGTH	DS	F	Length of source environment data
MQRMH_SRCENVOFFSET	DS	F	Offset of source environment data
MQRMH_SRCNAMELENGTH	DS	F	Length of source object name
MQRMH_SRCNAMEOFFSET	DS	F	Offset of source object name
MQRMH_DESTENVLENGTH	DS	F	Length of destination environment
*			data

MQRMH_DESTENVOFFSET	DS	F	Offset of destination environment data
*MQRMH_DESTNAMELENGTH	DS	F	Length of destination object name
MQRMH_DESTNAMEOFFSET	DS	F	Offset of destination object name
MQRMH_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)

Wizualna deklaracja podstawowa

```

Type MQRMH
  StructId      As String*4 'Structure identifier'
  Version       As Long     'Structure version number'
  StructLength  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-rekord odpowiedzi

W poniższej tabeli podsumowano pola w strukturze.

Tabela 82. Pola w MQRR		
Pole	Opis	Temat
CompCode	Kod zakończenia dla kolejki	CompCode
Reason	Kod przyczyny dla kolejki	Powód

Przegląd dla MQRR

Dostępność: klienci AIX, HP-UX, IBM i, Solaris, Linux, Windowsi IBM MQ połączone z tymi systemami.

Przeznaczenie: Użyj struktury MQRR, aby otrzymywać kod zakończenia i kod przyczyny wynikające z operacji otwierania lub umieszczania dla jednej kolejki docelowej, gdy miejscem docelowym jest lista dystrybucyjna. MQRR jest strukturą wyjściową dla wywołań MQOPEN, MQPUT i MQPUT1.

Zestaw znaków i kodowanie: Dane w tabeli MQRR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniach MQOPEN i MQPUT lub w wywołaniu MQPUT1, można określić kody zakończenia i kody przyczyny dla wszystkich kolejek na liście dystrybucyjnej, gdy wynik wywołania jest mieszany, to znaczy, gdy wywołanie powiedzie się dla niektórych kolejek na liście, ale dla innych nie powiedzie się. Kod przyczyny MQRC_MULTIPLE_UZASADNIENIE z wywołania wskazuje,

że rekordy odpowiedzi (o ile zostały udostępnione przez aplikację) zostały ustawione przez menedżer kolejek.

Pola dla tabeli MQRR

Struktura MQRR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CompCode (MQLONG)

Jest to kod zakończenia wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest MQCC_OK.

Przyczyna (MQLONG)

Jest to kod przyczyny wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest MQRC_NONE.

Wartości początkowe i deklaracje języków dla MQRR

Tabela 83. Początkowe wartości pól w MQRR dla MQRR		
Nazwa pola	Nazwa stałej	Wartość stałej
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
Uwagi: 1. W języku programowania C: zmienna makraParametr MQRR_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQRR MyRR = {MQRR_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQRR MQRR;  
struct tagMQRR {  
    MQLONG CompCode; /* Completion code for queue */  
    MQLONG Reason; /* Reason code for queue */  
};
```

Deklaracja języka COBOL

```
** MQRR structure  
10 MQRR.  
** Completion code for queue  
15 MQRR-COMPCODE PIC S9(9) BINARY.  
** Reason code for queue  
15 MQRR-REASON PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl  
1 MQRR based,  
3 CompCode fixed bin(31), /* Completion code for queue */  
3 Reason fixed bin(31); /* Reason code for queue */
```

```

Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type

```

MQSCO-opcje konfiguracji protokołu SSL

W poniższej tabeli podsumowano pola w strukturze.

Tabela 84. Zmienne w MQSCO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>KeyRepository</i>	Położenie repozytorium kluczy	KeyRepository
<i>CryptoHardware</i>	Szczegóły dotyczące sprzętu szyfrującego	CryptoHardware
<i>AuthInfoRecCount</i>	Liczba obecnych rekordów MQAIR	AuthInfoRecCount
<i>AuthInfoRecOffset</i>	Przesunięcie pierwszego rekordu MQAIR od początku wywołania MQSCO	AuthInfoRecOffset
<i>AuthInfoRecPtr</i>	Adres pierwszego rekordu MQAIR	AuthInfoRecPtr
Uwaga: Następujące dwa pola są ignorowane, jeśli <i>Version</i> jest mniejsze niż MQSCO_VERSION_2.		
<i>KeyResetCount</i>	Liczba resetowanych kluczy tajnych SSL	KeyResetLiczba
<i>FipsRequired</i>	Korzystanie z algorytmów szyfrowania z certyfikatem FIPS w produkcie IBM MQ	“FipsRequired (MQLONG)” na stronie 564
Uwaga: Następujące pole jest ignorowane, jeśli <i>Version</i> jest mniejsze niż MQSCO_VERSION_3.		
<i>EncryptionPolicySuiteB</i>	Używaj tylko algorytmów szyfrujących Suite B	EncryptionPolicySuiteB
Uwaga: Następujące pole jest ignorowane, jeśli <i>Version</i> jest mniejsze niż MQSCO_VERSION_4.		
<i>CertificateValPolicy</i>	Strategia sprawdzania poprawności certyfikatu	Strategia CertificateVal
Uwaga: Następujące pole jest ignorowane, jeśli <i>Version</i> jest mniejsze niż MQSCO_VERSION_5.		
<i>CertificateLabel</i>	Szczegóły etykiety certyfikatu, która jest używana.	CertificateLabel

Odsyłacze pokrewne

“MQCNO-opcje połączenia” na stronie 320

W poniższej tabeli podsumowano pola w strukturze.

“Przegląd produktu MQSCO” na stronie 561

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux and Windows clients.

“Pola dla MQSCO” na stronie 562

[“Wartości początkowe i deklaracje języków dla MQSCO” na stronie 566](#)

Przegląd produktu MQSCO

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux and Windows clients.

Cel: Struktura MQSCO (w połączeniu z polami SSL w strukturze MQCD) umożliwia uruchamianie aplikacji jako IBM MQ MQI client w celu określenia opcji konfiguracyjnych, które sterują użyciem protokołu SSL dla połączenia klienta, gdy protokołem kanału jest protokół TCP/IP. Struktura jest parametrem wejściowym w wywołaniu MQCONN.

Jeśli protokół kanału dla kanału klienta nie jest protokołem TCP/IP, struktura MQSCO jest ignorowana.

Zestaw znaków i kodowanie: Dane w tabeli MQSCO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu *CodedCharSetId* oraz kodowanie lokalnego menedżera kolejek nadawanego przez komendę MQENC_NATIVE.

Pola dla MQSCO

Struktura MQSCO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

AuthInfoRecCount (MQLONG)

Jest to liczba rekordów informacji uwierzytelniających (MQAIR), do których adresowane są pola *AuthInfoRecPtr* lub *AuthInfoRecOffset*. Więcej informacji na ten temat zawiera sekcja [“MQAIR-rekord informacji uwierzytelniającej”](#) na stronie 275. Wartość musi być równa zero lub większa. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_REC_COUNT_ERROR.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

AuthInfoRecOffset (MQLONG)

Jest to przesunięcie w bajtach pierwszego rekordu informacji uwierzytelniających od początku struktury MQSCO. Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli wartość *AuthInfoRecCount* wynosi zero.

Aby określić rekordy MQAIR, ale nie oba, można użyć opcji *AuthInfoRecOffset* lub *AuthInfoRecPtr*, aby uzyskać szczegółowe informacje, należy zapoznać się z opisem pola *AuthInfoRecPtr*.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

AuthInfoRecPtr (PMQAIR)

Jest to adres pierwszego rekordu informacji uwierzytelniających. Pole jest ignorowane, jeśli wartość *AuthInfoRecCount* wynosi zero.

Tablicę rekordów MQAIR można udostępnić na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *AuthInfoRecPtr*

W takim przypadku aplikacja może zadeklarować tablicę rekordów MQAIR, która jest oddzielona od struktury MQSCO, i ustawić parametr *AuthInfoRecPtr* na adres tablicy.

Należy rozważyć użycie produktu *AuthInfoRecPtr* dla języków programowania, które obsługują typ danych wskaźnika w sposób przenośny dla różnych środowisk (na przykład język programowania w języku C).

- Za pomocą pola przesunięcia *AuthInfoRecOffset*

W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą obiekt MQSCO, po którym następuje tablica rekordów MQAIR, a następnie ustawić parametr *AuthInfoRecOffset* na przesunięcie pierwszego rekordu w tablicy od początku struktury MQSCO. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być zakwaterowana w tabeli MQLONG (najbardziej restrykcyjnym językiem programowania jest język COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

Należy rozważyć użycie produktu *AuthInfoRecOffset* w językach programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który nie jest przenośny dla różnych środowisk (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki można użyć tylko jednej z następujących opcji: *AuthInfoRecPtr* i *AuthInfoRecOffset* ; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_REC_ERROR, jeśli oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

CertificateLabel (MQCHAR64)

W tym polu podano szczegółowe informacje na temat używanej etykiety certyfikatu.

Program IBM MQ inicjuje wartość domyślną dla pola *CertificateLabel* jako odstęp.

Wartość ta jest interpretowana w czasie wykonywania jako wartość domyślna i jest zgodna wstecznie.

Na przykład określenie wersji MQSCO mniejszej niż 5.0 lub użycie wartości domyślnej odstępów w polu *CertificateLabel* powoduje użycie wcześniej istniejącej wartości domyślnej *ibmwebsphermq<user_id>*.

Strategia CertificateVal(MQLONG)

To pole określa typ strategii sprawdzania poprawności certyfikatu. Pole można ustawić na jedną z następujących wartości:

MQ_CERT_VAL_POLICY_ANY

Zastosuj każdą ze strategii sprawdzania poprawności certyfikatów obsługiwanych przez bibliotekę bezpiecznych gniazd. Zaakceptuj łańcuch certyfikatów, jeśli dowolna z strategii uzna, że łańcuch certyfikatów jest poprawny.

MQ_CERT_VAL_POLICY_RFC5280

Zastosuj tylko strategię sprawdzania poprawności certyfikatu zgodną ze standardem RFC5280 . To ustawienie zapewnia bardziej restrykcyjne sprawdzanie poprawności niż ustawienie ANY, ale odrzuca niektóre starsze certyfikaty cyfrowe.

Początkowa wartość tego pola to MQ_CERT_VAL_POLICY_ANY

CryptoHardware (MQCHAR256)

To pole zawiera szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienckim.

Ustaw pole na łańcuch w następującym formacie lub pozostaw to pole puste lub puste:

```
GSK_PKCS11=< the PKCS #11 driver path and file name >;< the PKCS #11 token label >;< the PKCS #11 token password >; <symmetric cipher setting>;
```

Aby używać sprzętu szyfrującego zgodnego z interfejsem PKCS #11 , na przykład IBM 4960 lub IBM 4764, należy określić ścieżkę do sterownika PKCS #11 , etykietę znacznika PKCS #11 i łańcuchy haseł tokenu PKCS #11 , każde z nich zakończone znakiem średnika.

Ścieżka do sterownika PKCS #11 jest pełną ścieżką do biblioteki współużytkowanej udostępniających obsługę karty PKCS #11 . Nazwa pliku sterownika PKCS #11 jest nazwą biblioteki współużytkowanej. Przykładem wartości wymaganej dla ścieżki i #11 pliku #11 PKCS jest:

```
/usr/lib/pkcs11/PKCS11_API.so
```

Etykieta tokenu PKCS #11 musi być zgodna z etykietą, z którą skonfigurowano sprzęt.

Jeśli nie jest wymagana żadna konfiguracja sprzętu szyfrującego, należy ustawić pole puste lub mieć wartość NULL.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Jeśli wartość ta nie jest poprawna lub wystąpi błąd podczas konfigurowania sprzętu szyfrującego, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_CRYPTOHARDWARE_ERROR.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_SSL_CRYPTO_HARDWARE_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C oraz puste znaki w innych językach programowania.

EncryptionPolicySuiteB(MQLONG)

To pole określa, czy używana jest kryptografia zgodna ze standardem Suite B, oraz jaki poziom siły jest używany. Wartość może być jedną lub większą z następujących wartości:

- MQ_SUITE_B_NONE

Kryptografia zgodna z pakietem B nie jest używana.

- MQ_SUITE_B_128_BIT

Używane są 128-bitowe zabezpieczenie mocy 128-bitowe Suite.

- MQ_SUITE_B_192_BIT

Pakiet B 192-bit bezpieczeństwa mocy jest używany.

Uwaga: Użycie wartości MQ_SUITE_B_NONE z żadną inną wartością w tym polu jest niepoprawne.

FipsRequired (MQLONG)

Produkt IBM MQ może być skonfigurowany z użyciem sprzętu szyfrującego, dzięki czemu używane moduły szyfrowania są dostarczane przez produkt sprzętowy. Te moduły mogą być certyfikowane zgodnie ze standardem FIPS do konkretnego poziomu w zależności od używanego produktu sprzętowego. W tym polu można określić, że używane są tylko algorytmy z certyfikatem FIPS, jeśli kryptografia jest dostarczona w oprogramowaniu dostarczonym w produkcie IBM MQ.

Po zainstalowaniu produktu IBM MQ instalowana jest również implementacja szyfrowania SSL, która udostępnia moduły z certyfikatem FIPS.

Możliwe wartości to:

MQSSL_FIPS_NO

Jest to wartość domyślna. Po ustawieniu tej wartości:

- Można użyć dowolnego obiektu CipherSpec obsługiwanego na konkretnej platformie.
- W przypadku uruchamiania bez użycia sprzętu szyfrującego, następujące specyfikacje CipherSpecs są uruchamiane za pomocą szyfrowania z certyfikatem FIPS 140-2 na platformach IBM MQ :
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

W przypadku ustawienia tej wartości, o ile nie jest używany sprzęt szyfrujący do wykonania kryptografii, można mieć pewność, że

- W specyfikacji CipherSpec stosowane do tego połączenia klienta mogą być używane tylko algorytmy szyfrowania certyfikowane przez FIPS.
- Połączenia przychodzące i wychodzące kanału SSL są pomyślne tylko wtedy, gdy używany jest jeden z następujących specyfikacji szyfru:
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

Uwagi:

1. Klasa CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna.
2. Jeśli to możliwe, jeśli skonfigurowano tylko standard FIPS (CipherSpecs), klient MQI odrzuca połączenia, które określają atrybut CipherSpec inny niż FIPS przy użyciu parametru MQRC_SSL_INITIALIZATION_ERROR. Produkt IBM MQ nie gwarantuje odrzucenia wszystkich takich

połączeń i jest odpowiedzialny za określenie, czy konfiguracja produktu IBM MQ jest zgodna ze standardem FIPS.

KeyRepository (MQCHAR256)

To pole ma znaczenie tylko w przypadku produktu IBM MQ MQI clients działającego w systemach UNIX, Linux i Okna. Określa ono położenie pliku bazy danych kluczy, w którym są przechowywane klucze i certyfikaty. Plik bazy danych kluczy musi mieć nazwę pliku o nazwie `zzz.kdb`, gdzie `zzz` jest wybierany przez użytkownika. Pole *KeyRepository* zawiera ścieżkę do tego pliku wraz z trzonkiem nazwy pliku (wszystkie znaki w nazwie pliku, do których nie ma, ale nie zawiera finalnego `.kdb`). Przyrostek pliku `.kdb` jest dodawany automatycznie.

Każdy plik bazy danych kluczy ma powiązany *plik ukrytych haseł*. Zawiera zakodowane hasła, które umożliwiają programistyczny dostęp do bazy danych kluczy. Plik ukrytych haseł musi znajdować się w tym samym katalogu i mieć ten sam plik macierzysty, co baza danych kluczy, i musi kończyć się przyrostkiem `.sth`.

Na przykład, jeśli pole *KeyRepository* ma wartość `/xxx/yyy/key`, plikiem bazy danych kluczy musi być `/xxx/yyy/key.kdb`, a plikiem ukrytych haseł musi być `/xxx/yyy/key.sth`, gdzie `xxx` i `yyy` reprezentują nazwy katalogów.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Wartość nie jest sprawdzana. Jeśli wystąpił błąd podczas uzyskiwania dostępu do repozytorium kluczy, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny `MQRC_KEY_REPOSITORY_ERROR` (błąd: kod przyczyny `MQRC_KEY_REPOSITORY_ERROR`).

Aby uruchomić połączenie SSL z IBM MQ MQI client, należy ustawić *KeyRepository* na poprawną nazwę pliku bazy danych kluczy.

To jest pole wejściowe. Długość tego pola jest podana w tabeli `MQ_SSL_KEY_REPOSITORY_LENGTH`. Wartością początkową tego pola jest łańcuch o wartości `NULL` w języku C oraz puste znaki w innych językach programowania.

Licznik KeyReset(MQLONG)

Reprezentuje łączną liczbę niezasyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL lub TLS, zanim klucz tajny zostanie renegotjowany.

Liczba bajtów obejmuje informacje sterujące wysłane przez agenta MCA.

W przypadku określenia wartości resetowania klucza tajnego SSL lub TLS w zakresie od 1 bajtu do 32 kB, w kanałach SSL lub TLS zostanie użyta wartość klucza tajnego resetowania klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernych resetów klucza, które nastąpiłyby w przypadku małych wartości resetowania klucza tajnego SSL lub TLS.

To jest pole wejściowe. Wartość jest liczbą z zakresu od 0 do 999 999 999, z wartością domyślną równą 0. Należy użyć wartości 0, aby wskazać, że klucze tajne nigdy nie są renegotjowane.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

Identyfikator MQSCO_STRUC_ID

Identyfikator struktury opcji konfiguracji protokołu SSL.

Dla języka programowania C zdefiniowana jest także stała `MQSCO_STRUC_ID_ARRAY`; ma taką samą wartość jak `MQSCO_STRUC_ID`, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to `MQSCO_STRUC_ID`.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQSCO_VERSION_1

Struktura opcji konfiguracji protokołu SSL w wersji Version-1.

MQSCO_VERSION_2

Struktura opcji konfiguracji protokołu SSL w wersji Version-2 .

MQSCO_VERSION_3

Struktura opcji konfiguracji protokołu SSL w wersji Version-3 .

MQSCO_VERSION_4

Struktura opcji konfiguracji protokołu SSL w wersji Version-4 .

MQSCO_VERSION_5

Struktura opcji konfiguracji protokołu SSL w wersji Version-5 .

Następująca stała określa numer wersji bieżącej wersji:

MQSCO_CURRENT_VERSION

Bieżąca wersja struktury opcji konfiguracji SSL.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQSCO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQSCO

Tabela 85. Wartości początkowe pól w produkcji MQSCO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQSCO_STRUC_ID	'SC0~'
<i>Version</i>	MQSCO_CURRENT_VERSION	1
<i>KeyRepository</i>	Brak	Pusty łańcuch lub odstępy
<i>CryptoHardware</i>	Brak	Pusty łańcuch lub odstępy
<i>AuthInfoRecCount</i>	Brak	0
<i>AuthInfoRecOffset</i>	Brak	0
<i>AuthInfoRecPtr</i>	Brak	Pusty wskaźnik lub zerowe bajty
<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

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraMQSCO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Deklaracja C

```

typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
    MQCHAR4      StrucId;          /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQCHAR256    KeyRepository;    /* Location of SSL key */
                                   /* repository */
    MQCHAR256    CryptoHardware;   /* Cryptographic hardware */
                                   /* configuration string */
    MQLONG       AuthInfoRecCount; /* Number of MQAIR records */
                                   /* present */
    MQLONG       AuthInfoRecOffset; /* Offset of first MQAIR */
                                   /* record from start of */
                                   /* MQSCO structure */
    PMQAIR       AuthInfoRecPtr;   /* Address of first MQAIR */
                                   /* record */

/* Ver:1 */
    MQLONG       KeyResetCount;    /* Number of unencrypted */
                                   /* bytes sent/received */
                                   /* before secret key is */
                                   /* reset */
    MQLONG       FipsRequired;     /* Using FIPS-certified */
                                   /* algorithms */
/* Ver:2 */
    MQLONG       EncryptionPolicySuiteB[4]; /* Use only Suite B */
/* Ver:3 */
    MQLONG       CertificateValPolicy; /* cryptographic algorithms */
                                   /* Certificate validation */
                                   /* policy */
/* Ver:4 */
    MQCHAR64     CertificateLabel; /* Certificate label */
/* Ver:5 */

```

Deklaracja języka COBOL

```

** MQSCO structure
10 MQSCO.
** Structure identifier
15 MQSCO-STRUCID PIC X(4).
** Structure version number
15 MQSCO-VERSION PIC S9(9) BINARY.
** Location of SSL key repository
15 MQSCO-KEYREPOSITORY PIC X(256).
** Cryptographic hardware configuration string
15 MQSCO-CRYPTOHARDWARE PIC X(256).
** Number of MQAIR records present
15 MQSCO-AUTHINFORECCOUNT PIC S9(9) BINARY.
** Offset of first MQAIR record from start of MQSCO structure
15 MQSCO-AUTHINFORECOFFSET PIC S9(9) BINARY.
** Address of first MQAIR record
15 MQSCO-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

```

Deklaracja PL/I

```

dcl
1 MQSCO based,
3 StrucId char(4), /* Structure identifier */
3 Version fixed bin(31), /* Structure version number */
3 KeyRepository char(256), /* Location of SSL key
                             repository */
3 CryptoHardware char(256), /* Cryptographic hardware
                             configuration string */

```

```

3 AuthInfoRecCount      fixed bin(31), /* Number of MQAIR records
                        present */
3 AuthInfoRecOffset     fixed bin(31), /* Offset of first MQAIR record
                        from start of MQSCO structure */
3 AuthInfoRecPtr        pointer,      /* Address of first MQAIR record */
3 KeyResetCount         fixed bin(31), /* Key reset count */
/* Version 1 */
3 FipsRequired          fixed bin(31), /* FIPS required */
/* Version 2 */
3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
3 CertificateValPolicy   fixed bin(31); /* Certificate validation policy */
/* Version 4 */

```

Wizualna deklaracja podstawowa

```

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'
  AuthInfoRecCount As Long   'Number of MQAIR records present'
  AuthInfoRecOffset As Long   'Offset of first MQAIR record from'
  AuthInfoRecPtr As MQPTR    'start of MQSCO structure'
  KeyResetCount As Long      'Address of first MQAIR record'
  is reset'
  'Version 1'
  FipsRequired As Long      'Number of unencrypted bytes sent/received before secret key
  'Version 2'
End Type

```

MQSD-deskryptor subskrypcji

W poniższej tabeli podsumowano pola w strukturze.

Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje
<i>ObjectName</i>	Nazwa obiektu	ObjectName
<i>AlternateUserId</i>	Alternatywne ID użytkownika	IdentyfikatorAlternateUser
<i>AlternateSecurityId</i>	Alternatywny identyfikator zabezpieczeń	IdentyfikatorAlternateSecurity
<i>SubExpiry</i>	Utrata ważności subskrypcji	SubExpiry
<i>ObjectString</i>	Łańcuch obiektu	ObjectString
<i>SubName</i>	Nazwa subskrypcji	SubName
<i>SubUserData</i>	Dane użytkownika subskrypcji	DaneSubUser
<i>SubCorrelId</i>	Identyfikator korelacji subskrypcji	IdentyfikatorSubCorrel
<i>PubPriority</i>	Priorytet publikacji	PubPriority
<i>PubAccountingToken</i>	Znacznik rozliczania publikacji	PubAccountingToken
<i>PubAppIdentityData</i>	Dane tożsamości aplikacji publikacyjnych	PubAppIdentityData
<i>SelectionString</i>	Łańcuch zawierający kryteria wyboru	SelectionString
<i>SubLevel</i>	Poziom subskrypcji	SubLevel

Pole	Opis	Temat
<i>ResObjectString</i>	Długa nazwa obiektu	ResObjectŁącuch

Przegląd produktu MQSD

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS, plus IBM MQ MQI clients połączone z tymi systemami.

Przeznaczenie: Struktura MQSD służy do określania szczegółów dotyczących dokonanej subskrypcji.

Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSUB. Więcej informacji na ten temat zawiera sekcja [Uwagi dotyczące używania produktu MQSUB](#).

Zarządzane subskrypcje: Jeśli aplikacja nie ma konkretnej potrzeby użycia określonej kolejki jako miejsca docelowego dla tych publikacji, które są zgodne z jej subskrypcją, może ona korzystać z funkcji subskrypcji zarządzanej. Jeśli aplikacja zdecyduje się na korzystanie z subskrypcji zarządzanej, menedżer kolejek informuje subskrybenta o miejscu docelowym, w którym są wysyłane opublikowane komunikaty, udostępniając uchwyt obiektu jako wyjście wywołania MQSUB. Więcej informacji na ten temat zawiera sekcja [Hobj \(MQHOBJ\)-wejście/wyjście](#).

Po usunięciu subskrypcji menedżer kolejek podejmuje również działania w celu wyczyszczenia komunikatów, które nie zostały pobrane z zarządzanego miejsca docelowego, w następujących sytuacjach:

- Po usunięciu subskrypcji-za pomocą komendy MQCLOSE z opcją MQCO_REMOVE_SUB-i zarządzany Hobj jest zamknięty.
- Niejawne oznacza, że po utracie połączenia z aplikacją korzystała z nietrwałej subskrypcji (MQSO_NON_DURABLE)
- Po utracie ważności subskrypcji, ponieważ utraciła ważność, a zarządzany Hobj jest zamknięty.

Należy używać subskrypcji zarządzanych z subskrypcjami nietrwałymi, tak aby ta procedura czyszczenia mogła wystąpić, a komunikaty dotyczące zamkniętych subskrypcji nietrwałych nie zajmą miejsca w menedżerze kolejek. Trwałe subskrypcje mogą również używać zarządzanych miejsc docelowych.

Wersja: Bieżąca wersja MQSD to MQSD_VERSION_1.

Zestaw znaków i kodowanie: Dane w tabeli MQSD muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQSD

Struktura MQSD zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

Identyfikator AlternateSecurity(MQBYTE40)

Jest to identyfikator zabezpieczeń, który jest przekazywany z identyfikatorem AlternateUserdo usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji.

AlternateSecurityId jest używany tylko wtedy, gdy określono wartość MQSO_ALTERNATE_USER_AUTHORITY, a pole AlternateUserID nie jest całkowicie puste w stosunku do pierwszego znaku o kodzie zero lub do końca pola.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole nie zmienia się.

Więcej informacji na ten temat zawiera opis produktu ["Identyfikator AlternateSecurity\(MQBYTE40\)" na stronie 481](#) w typie danych MQOD.

Identyfikator AlternateUser(MQCHAR12)

Jeśli zostanie określona wartość MQSO_ALTERNATE_USER_AUTHORITY, to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla subskrypcji i dla danych wyjściowych w kolejce docelowej (określonej w parametrze *Hobj* wywołania MQSUB) zamiast identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli operacja się powiedzie, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji w miejscu identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli określono wartość MQSO_ALTERNATE_USER_AUTHORITY, a pole to jest całkowicie puste, do pierwszego znaku o kodzie zero lub do końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy nie jest wymagana autoryzacja użytkownika w celu zasubskrybowania tego tematu przy użyciu podanych opcji lub kolejki docelowej dla danych wyjściowych.

Jeśli wartość MQSO_ALTERNATE_USER_AUTHORITY nie jest określona, to pole jest ignorowane.

W podanych środowiskach istnieją następujące różnice:

- W systemie z/OS do sprawdzania autoryzacji dla subskrypcji używane są tylko pierwsze 8 znaków identyfikatora AlternateUser. Jednak bieżący identyfikator użytkownika musi być autoryzowany do określenia tego konkretnego alternatywnego identyfikatora użytkownika; dla tego sprawdzenia używane są wszystkie 12 znaków alternatywnego identyfikatora użytkownika. Identyfikator użytkownika musi zawierać tylko znaki dozwolone przez zewnętrznego menedżera zabezpieczeń.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole nie zmienia się.

To jest pole wejściowe. Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C, a 12 pustych znaków w innych językach programowania.

ObjectName (MQCHAR48)

Jest to nazwa obiektu tematu zgodnie z definicją w lokalnym menedżerze kolejek.

Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (od A do Z)
- Małe litery alfabetu (od a do z)
- Cyfry cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Użyj znaku o kodzie zero, aby wskazać koniec znaczących danych w nazwie; wartość NULL i wszystkie znaki po nim są traktowane jako znaki puste. W środowiskach wskazanych poniżej obowiązują następujące ograniczenia:

- W systemach, w których używane jest kodowanie EBCDIC Katakana, nie można używać małych liter.
- wł.z/OS:
 - Należy unikać nazw, które rozpoczynają się lub kończą znakiem podkreślenia. Nie mogą one być przetwarzane przez panele kontrolne i operacyjne.
 - Znak procentu ma specjalne znaczenie dla RACF. Jeśli produkt RACF jest używany jako zewnętrzny menedżer zabezpieczeń, nazwy nie mogą zawierać wartości procentowej. W takim przypadku nazwy te nie są uwzględniane podczas sprawdzania zabezpieczeń, gdy używane są profile ogólne produktu RACF.
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w znaki cudzysłowu, gdy są określone w komendach. Tych znaków cudzysłowu nie należy określać dla nazw, które występują jako pola w strukturach lub jako parametry wywołań.

ObjectName jest używany do tworzenia pełnej nazwy tematu.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

If the object identified by the *ObjectName* field cannot be found, the call fails with reason code MQRC_UNKNOWN_OBJECT_NAME even if there is a string specified in *ObjectString*.

W przypadku powrotu z wywołania MQSUB przy użyciu opcji MQSO_RESUME to pole nie zmienia się.

Długość tego pola jest podana przez wartość MQ_TOPIC_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER nazwa subskrybowanego obiektu tematu nie może zostać zmieniona. To pole i pole *ObjectString* można pominąć. Jeśli są one udostępniane, muszą one zostać rozstrzygane do tej samej pełnej nazwy tematu. Jeśli nie, wywołanie zakończy się niepowodzeniem z błędem MQRC_TOPIC_NOT_ALTERABLE.

ObjectString (MQCHARV)

Jest to długa nazwa obiektu, która ma być używana.

ObjectString jest używany do tworzenia pełnej nazwy tematu.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

Maksymalna długość *ObjectString* wynosi 10240.

Jeśli wartość *ObjectString* nie została określona poprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_STRING_ERROR.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQCHARV.

Jeśli w *ObjectString* znajdują się znaki wieloznaczne, interpretacja tych znaków wieloznacznych może być sterowana za pomocą opcji Wildcard określonych w polu Opcje w MQSD.

W przypadku powrotu z wywołania MQSUB przy użyciu opcji MQSO_RESUME to pole nie zmienia się. Jeśli podano bufor, w polu *ResObjectString* zwracana jest pełna nazwa tematu.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, nie można zmienić długiej nazwy obiektu tematu zasubskrybowanego do subskrybowanego. To pole i pole *ObjectName* można pominąć. Jeśli są one udostępniane, muszą one zostać rozstrzygane do tej samej pełnej nazwy tematu lub wywołania nie powiedzie się z użyciem parametru MQRC_TOPIC_NOT_ALTERABLE.

Opcje (MQLONG)

Udostępnia opcje umożliwiające sterowanie działaniem wywołania MQSUB.

Należy określić co najmniej jedną z następujących opcji:

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Kombinacje, które nie są poprawne, są oznaczone w tym temacie; wszystkie pozostałe kombinacje są poprawne.

Opcje dostępu lub tworzenia: opcje dostępu i tworzenia kontrolują, czy subskrypcja została utworzona, czy też istniejąca subskrypcja jest zwracana, czy zmieniana. Należy określić co najmniej jedną z tych opcji. W tabeli wyświetlane są poprawne kombinacje opcji dostępu i tworzenia.

Kombinacja opcji	Uwagi
MQSO_CREATE	Tworzy subskrypcję, jeśli nie istnieje. Ta kombinacja nie powiedzie się, jeśli subskrypcja istnieje.
MQSO_RESUME	Wznawia istniejącą subskrypcję. Ta kombinacja nie powiedzie się, jeśli nie istnieje subskrypcja.
MQSO_CREATE + MQSO_RESUME	Tworzy subskrypcję, jeśli nie istnieje, a następnie wznawia dopasowanie, jeśli istnieje. Ta kombinacja jest przydatna, gdy jest używana w aplikacji, która jest uruchamiana pewną liczbę razy.
MQSO_ALTER (patrz uwaga)	Wznawia istniejącą subskrypcję, modyfikując wszystkie pola, które mają być zgodne z określonymi w MQSD. Ta kombinacja nie powiedzie się, jeśli nie istnieje subskrypcja.
MQSO_CREATE + MQSO_ALTER (patrz uwaga)	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje, a następnie wznawia zgodną jedną z nich, jeśli taka istnieje, zmieniając dowolne pola tak, aby były zgodne z tym, które zostały określone w MQSD. Ta kombinacja jest przydatna w przypadku użycia w aplikacji, która chce zapewnić, że jej subskrypcja znajduje się w określonym stanie przed kontynuowaniem.

Uwaga:

Opcje określające wartość MQSO_ALTER mogą również określać wartość MQSO_RESUME, ale ta kombinacja nie ma dodatkowego wpływu na określenie parametru MQSO_ALTER w monoterapii. Komenda MQSO_ALTER implikuje wartość MQSO_RESUME, ponieważ wywołanie MQSUB w celu zmodyfikowania subskrypcji oznacza, że subskrypcja również zostanie wznowiona. Odwrotność nie jest jednak prawdą, jednak: wznawianie subskrypcji nie oznacza, że ma być ona zmieniona.

MQSO_CREATE

Utwórz nową subskrypcję dla określonego tematu. Jeśli istnieje subskrypcja z użyciem tej samej partycji *SubName*, wywołanie kończy się niepowodzeniem z opcją MQRC_SUB_ALREADY_EXISTS. Tego niepowodzenia można uniknąć, łącząc opcję MQSO_CREATE z opcją MQSO_RESUME. *SubName* nie zawsze jest konieczne. Więcej informacji na ten temat zawiera opis tego pola.

Połączenie MQSO_CREATE z opcją MQSO_RESUME zwraca uchwyt do istniejącej subskrypcji dla określonego *SubName*, jeśli zostanie znaleziony. Jeśli nie istnieje subskrypcja, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól dostępnych w MQSD.

Komenda MQSO_CREATE może być również połączona z działaniem MQSO_ALTER w celu wykonania podobnego działania.

MQSO_RESUME

Zwraca uchwyt do wcześniej istniejącej subskrypcji, która jest zgodna z nazwą podaną w polu *SubName*. Żadne zmiany nie są wprowadzane do zgodnych atrybutów subskrypcji i są one zwracane w wyniku w strukturze MQSD. Używane są tylko następujące pola MQSD: StrucId, Version, Options, AlternateUserId i AlternateSecurityId i SubName.

Wywołanie nie powiodło się z kodem przyczyny MQRC_NO_SUBSCRIPTION, jeśli subskrypcja nie istnieje, a jej nazwa jest zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję MQSO_CREATE z opcją MQSO_RESUME.

ID użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika z ostatniej pomyślnej zmiany. Jeśli używany jest identyfikator AlternateUser, a dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, alternatywny identyfikator

użytkownika jest rejestrowany jako ID użytkownika, który utworzył subskrypcję, a nie ID użytkownika, pod którym subskrypcja została wykonana.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji MQSO_ANY_USERID, a identyfikator użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu do subskrypcji, wywołanie nie powiedzie się i zostanie zakodowany kod przyczyny MQRC_IDENTITY_MISMATCH.

Jeśli zgodna subskrypcja istnieje i jest aktualnie używana, wywołanie nie powiedzie się i zostanie użyta wartość MQRC_SUBSCRIPTION_IN_USE.

Jeśli subskrypcja o nazwie SubName nie jest poprawną subskrypcją w celu wznowienia lub zmiany aplikacji, wywołanie nie powiedzie się i zostanie zakończona subskrypcja MQRC_INVALID_SUBSCRIPTION (MQRC_INVALID_SUBSCRIPTION).

Komenda MQSO_RESUME jest niejawna przy użyciu komendy MQSO_ALTER, dlatego nie ma potrzeby łączenia jej z tą opcją. Jednak połączenie tych dwóch opcji nie powoduje błędu.

MQSO_ALTER

Zwraca uchwyt do wcześniejszej subskrypcji z pełną nazwą subskrypcji zgodną z nazwą podaną w polu *SubName*. Wszystkie atrybuty subskrypcji, które są inne niż te określone w tabeli MQSD, są zmieniane w subskrypcji, chyba że zmiana jest niedozwolona dla tego atrybutu. Szczegóły znajdują się w opisie każdego atrybutu i są podsumowane w poniższej tabeli. W przypadku próby zmiany atrybutu, którego nie można zmienić, lub zmiany subskrypcji, która ustawiła opcję MQSO_IMMUTABLE, wywołanie nie powiedzie się i zostanie wyświetlony kod przyczyny przedstawiony w poniższej tabeli.

Wywołanie nie powiodło się z kodem przyczyny MQRC_NO_SUBSCRIPTION, jeśli subskrypcja zgodna z pełną nazwą subskrypcji nie istnieje. Tego niepowodzenia można uniknąć, łącząc opcję MQSO_CREATE z MQSO_ALTER.

Połączenie MQSO_CREATE z MQSO_ALTER zwraca uchwyt do istniejącej subskrypcji dla określonego *SubName*, jeśli zostanie znaleziony. Jeśli nie istnieje subskrypcja, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól dostępnych w MQSD.

ID użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli jest on później modyfikowany przez inny identyfikator użytkownika, jest to identyfikator użytkownika z ostatniej, pomyślnej zmiany. Jeśli używany jest identyfikator AlternateUser, a dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, to alternatywny identyfikator użytkownika jest rejestrowany jako ID użytkownika, który utworzył subskrypcję, a nie ID użytkownika, pod którym subskrypcja została wykonana.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji MQSO_ANY_USERID, a ID użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu do subskrypcji, wywołanie nie powiedzie się, kod przyczyny MQRC_IDENTITY_MISMATCH.

Jeśli zgodna subskrypcja istnieje i jest aktualnie używana, wywołanie nie powiedzie się i zostanie użyta wartość MQRC_SUBSCRIPTION_IN_USE.

Jeśli subskrypcja o nazwie SubName nie jest poprawną subskrypcją w celu wznowienia lub zmiany aplikacji, wywołanie nie powiedzie się i zostanie zakończona subskrypcja MQRC_INVALID_SUBSCRIPTION (MQRC_INVALID_SUBSCRIPTION).

W poniższej tabeli przedstawiono możliwość zmiany wartości atrybutów w MQSD i MQSUB za pomocą komendy MQSO_ALTER.

Tabela 86. Atrybuty w tabelach MQSD i MQSUB, które mogą być zmieniane			
Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Czy ten atrybut może zostać zmieniony za pomocą komendy MQSO_ALTER	Kod przyczyny
MQSD	Opcje trwałości	Nie	MQRC_DURABILITY_NOT_ALTERABLE
MQSD	Opcje miejsca docelowego	Tak	Brak

Tabela 86. Atrybuty w tabelach MQSD i MQSUB, które mogą być zmieniane (kontynuacja)

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Czy ten atrybut może zostać zmieniony za pomocą komendy MQSO ALTER	Kod przyczyny
MQSD	Opcje rejestracji	Tak (patrz uwaga "1" na stronie 574)	MQRC_GROUPING_NOT_ALTERABLE-jeśli próbujesz zmienić wartość MQSO_GROUP_SUB
MQSD	Opcje publikacji	Tak (patrz uwaga "2" na stronie 574)	Brak
MQSD	Opcje wieloznaczne	Nie	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Inne opcje	Nie (patrz uwaga "3" na stronie 574)	Brak
MQSD	ObjectName	Nie	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Identyfikator AlternateUser	Nie (patrz uwaga "4" na stronie 574)	Brak
MQSD	Identyfikator AlternateSecurity	Nie (patrz uwaga "4" na stronie 574)	Brak
MQSD	SubExpiry	Tak	Brak
MQSD	ObjectString	Nie	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	Nie (patrz uwaga "5" na stronie 574)	Brak
MQSD	Dane SubUser	Tak	Brak
MQSD	Identyfikator SubCorrel	Tak (patrz uwaga "6" na stronie 574)	MQRC_GROUPING_NOT_ALTERABLE, gdy w grupie subskrypcji
MQSD	PubPriority	Tak	Brak
MQSD	Znacznik PubAccounting	Tak	Brak
MQSD	PubApplIdentityData	Tak	Brak
MQSD	SubLevel	Nie	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	Hobj	Tak (patrz uwaga "6" na stronie 574)	MQRC_GROUPING_NOT_ALTERABLE, gdy w grupie subskrypcji

Uwagi:

1. Nie można zmienić opcji MQSO_GROUP_SUB.
2. MQSO_NEW_PUBLICATIONS_ONLY nie może zostać zmienione, ponieważ nie jest częścią subskrypcji
3. Te opcje nie są częścią subskrypcji
4. Ten atrybut nie jest częścią subskrypcji
5. Ten atrybut jest tożsamością modyfikowanej subskrypcji
6. Możliwa do zmiany, z wyjątkiem sytuacji, gdy część pogrupowanej subskrypcji (MQSO_GROUP_SUB)

Opcje trwałości: Następujące opcje sterują sposobem, w jaki jest trwała subskrypcja. Można określić tylko jedną z tych opcji. Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji MQSO ALTER, nie można zmienić trwałości subskrypcji. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME, ustawiona jest odpowiednia opcja trwałości.

MQSO_DURABLE,

Zażądaj, aby subskrypcja tego tematu pozostała do czasu jawnego usunięcia za pomocą komendy MQCLOSE z opcją MQCO_REMOVE_SUB. Jeśli ta subskrypcja nie zostanie jawnie usunięta, pozostanie ona nawet po zamknięciu połączenia aplikacji z menedżerem kolejek.

Jeśli zażądano trwałej subskrypcji do tematu, który jest zdefiniowany jako nie zezwalający na trwałe subskrypcje, wywołanie kończy się niepowodzeniem z opcją MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NON_DURABLE,

Żądanie subskrypcji tego tematu zostanie usunięte, gdy połączenie aplikacji z menedżerem kolejek zostanie zamknięte, jeśli nie zostało jeszcze jawnie usunięte. MQSO_NON_DURABLE

jest przeciwieństwem opcji MQSO_DURABLE i jest ona zdefiniowana w dokumentacji programu pomocowego. Jest to wartość domyślna, jeśli nie zostanie podana żadna wartość.

Opcje docelowe: Poniższa opcja steruje miejscem docelowym, do którego są wysyłane publikacje dotyczące subskrybowanego tematu. W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, można zmienić miejsce docelowe używane na potrzeby publikacji dotyczących subskrypcji. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME ta opcja jest ustawiana, jeśli jest odpowiednia.

MQSO_MANAGED

Zažadaj, aby miejsce docelowe, do którego są wysyłane publikacje, jest zarządzane przez menedżer kolejek.

Uchwyt obiektu zwrócony w produkcie *Hobj* reprezentuje kolejkę zarządzaną menedżera kolejek i jest używany z kolejnymi wywołaniami MQGET, MQCB, MQINQ lub MQCLOSE.

Uchwyt obiektu zwrócony z poprzedniego wywołania MQSUB nie może być podany w parametrze *Hobj*, jeśli nie określono parametru MQSO_MANAGED.

MQSO_NO_MULTICAST

Zažadaj, aby miejsce docelowe, do którego wysyłane są publikacje, nie jest adresem grupowym rozsyłania grupowego. Ta opcja jest poprawna tylko wtedy, gdy jest połączona z opcją MQSO_MANAGED. Jeśli uchwyt do kolejki jest podany w parametrze *Hobj*, nie można użyć rozsyłania grupowego dla tej subskrypcji, a opcja nie jest poprawna.

Jeśli temat został zdefiniowany w taki sposób, aby zezwalać na subskrypcje rozsyłania grupowego, należy użyć ustawienia MCAST (ONLY), a następnie wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MULTICAST_REQUIRED.

Opcja zasięgu: Poniższa opcja określa zasięg subskrybowanego subskrypcji. W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER ta opcja zasięgu subskrypcji nie może zostać zmieniona. Po powrocie z wywołania MQSUB za pomocą komendy MQSO_RESUME, ustawiona jest odpowiednia opcja zasięgu.

MQSO_SCOPE_QMGR

Ta subskrypcja jest dokonywana tylko w lokalnym menedżerze kolejek. Żadna subskrypcja proxy nie jest dystrybuowana do innych menedżerów kolejek w sieci. Do tego subskrybenta są wysyłane tylko te publikacje, które są publikowane w tym menedżerze kolejek. Spowoduje to przesłonięcie dowolnego zestawu zachowań za pomocą atrybutu tematu SUBSCOPE.

Uwaga: Jeśli ta opcja nie zostanie ustawiona, zasięg subskrypcji jest określany na podstawie atrybutu tematu SUBSCOPE.

Opcje rejestracji: Następujące opcje sterują szczegółami rejestracji, które są wykonywane w menedżerze kolejek dla tej subskrypcji. W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER te opcje rejestracji mogą zostać zmienione. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME, ustawione są odpowiednie opcje rejestracji.

MQSO_GROUP_SUB

Ta subskrypcja ma być pogrupowana z innymi subskrypcjami tego samego SubLevel przy użyciu tej samej kolejki i określaniem tego samego identyfikatora korelacji, tak aby wszystkie publikacje dotyczące tematów, które spowodowałyby, że grupa subskrypcji była udostępniana grupie subskrypcji, ze względu na nakładający się zestaw łańcuchów tematów używanych, powoduje, że tylko jeden komunikat jest dostarczany do kolejki. Jeśli ta opcja nie jest używana, każda unikalna subskrypcja (identyfikowana przez SubName), która jest zgodna, jest udostępniona z kopią publikacji, która może oznaczać więcej niż jedną kopię publikacji, która może być umieszczona w kolejce współużytkowanej przez liczbę subskrypcji.

Tylko najbardziej znacząca subskrypcja w grupie jest dostarczona z kopią publikacji. Najbardziej znacząca subskrypcja jest oparta na pełnej nazwie tematu aż do punktu, w którym znajduje się znak wieloznaczny. Jeśli w obrębie grupy używana jest mieszanka schematów wieloznacznych, ważna

jest tylko pozycja znaku wieloznacznego. Zaleca się, aby nie łączyć różnych schematów znaków wieloznacznych w ramach grupy subskrypcji, które współużytkują tę samą kolejkę.

Podczas tworzenia nowej zgrupowanej subskrypcji musi on mieć jeszcze unikalną nazwę SubName, ale jeśli jest ona zgodna z pełną nazwą tematu istniejącej subskrypcji w grupie, wywołanie kończy się niepowodzeniem z opcją MQRC_DUPLICATE_GROUP_SUB.

Jeśli najbardziej znacząca subskrypcja grupy określa także wartość MQSO_NOT_OWN_PUBS i jest to publikacja pochodząca z tej samej aplikacji, żadna publikacja nie zostanie dostarczona do kolejki.

W przypadku zmiany subskrypcji z tą opcją, pola, które oznaczają grupowanie, Hobj w wywołaniu MQSUB (reprezentujące kolejkę i nazwę menedżera kolejek) oraz identyfikator SubCorrelnie mogą zostać zmienione. Próba ich zmiany powoduje, że wywołanie nie powiodło się. MQRC_GROUPING_NOT_ALTERABLE nie jest możliwe.

Ta opcja musi być połączona z identyfikatorem MQSO_SET_CORREL_ID o identyfikatorze SubCorrel, który nie jest ustawiony na wartość MQCI_NONE i nie może być łączony z MQSO_MANAGED.

MQSO_ANY_USERID

Jeśli określono atrybut MQSO_ANY_USERID, tożsamość subskrybenta nie jest ograniczona do pojedynczego identyfikatora użytkownika. Dzięki temu każdy użytkownik może zmienić lub wznowić subskrypcję, gdy mają odpowiednie uprawnienia. Abonament może mieć tylko jeden użytkownik w dowolnym momencie. Próba wznowienia użycia subskrypcji, która jest obecnie używana przez inną aplikację, powoduje niepowodzenie wywołania z parametrem MQRC_SUBSCRIPTION_IN_USE.

Aby dodać tę opcję do istniejącej subskrypcji, wywołanie MQSUB (za pomocą komendy MQSO_ALTER) musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli wywołanie MQSUB odwołuje się do istniejącej subskrypcji z ustawioną nazwą MQSO_ANY_USERID, a identyfikator użytkownika różni się od oryginalnej subskrypcji, wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienia do subskrybowania tematu. Po pomyślnym zakończeniu, przyszłe publikacje tego subskrybenta są umieszczane w kolejce subskrybentów z nowym identyfikatorem użytkownika ustawionym w komunikacie publikacji.

Nie określaj jednocześnie wartości MQSO_ANY_USERID i MQSO_FIXED_USERID. Jeśli nie zostanie podana żadna wartość, domyślną wartością jest MQSO_FIXED_USERID.

MQSO_FIXED_USERID

Jeśli określono atrybut MQSO_FIXED_USERID, subskrypcja może zostać zmieniona lub wznowiona tylko przez ostatni identyfikator użytkownika w celu zmiany subskrypcji. Jeśli subskrypcja nie została zmieniona, jest to identyfikator użytkownika, który utworzył subskrypcję.

Jeśli komenda MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością MQSO_ANY_USERID i zmienia subskrypcję za pomocą komendy MQSO_ALTER w celu użycia opcji MQSO_FIXED_USERID, to identyfikator użytkownika subskrypcji jest teraz stały przy użyciu tego nowego identyfikatora użytkownika. Wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienia do subskrybowania tematu.

Jeśli identyfikator użytkownika inny niż zarejestrowany jako będący właścicielem subskrypcji próbuje wznowić lub zmienić subskrypcję MQSO_FIXED_USERID, wywołanie kończy się niepowodzeniem z błędem MQRC_IDENTITY_MISMATCH. Identyfikator użytkownika będącego właścicielem subskrypcji można wyświetlić za pomocą komendy DISPLAY SBSTATUS.

Nie określaj jednocześnie wartości MQSO_ANY_USERID i MQSO_FIXED_USERID. Jeśli nie zostanie podana żadna wartość, domyślną wartością jest MQSO_FIXED_USERID.

Opcje publikacji: Następujące opcje sterują sposobem wysyłania publikacji do tego subskrybenta. W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER opcje te mogą zostać zmienione.

MQSO_NOT_OWN_PUBS

Informuje brokera o tym, że aplikacja nie chce wyświetlać żadnych własnych publikacji. Publikacje są uznawane za pochodzące z tej samej aplikacji, jeśli uchwyt połączeń są takie same. W przypadku

powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME ta opcja jest ustawiana, jeśli jest odpowiednia.

MQSO_NEW_PUBLICATIONS_ONLY (TYLKO)

Obecnie nie są wysyłane publikacje, które mają być wysyłane, gdy ta subskrypcja jest tworzona, a tylko nowe publikacje. Ta opcja ma zastosowanie tylko wtedy, gdy określona jest wartość MQSO_CREATE. Wszelkie późniejsze zmiany w subskrypcji nie wpływają na przepływ publikacji, a więc wszelkie publikacje zachowane na danym temacie, będą już wysyłane do subskrybenta jako nowe publikacje.

Jeśli ta opcja zostanie podana bez wywołania MQSO_CREATE, wywołanie nie powiedzie się i zostanie użyta wartość MQRC_OPTIONS_ERROR. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME opcja ta nie jest ustawiona, nawet jeśli subskrypcja została utworzona przy użyciu tej opcji.

Jeśli ta opcja nie jest używana, poprzednio zachowane komunikaty są wysyłane do podanej kolejki docelowej. Jeśli to działanie nie powiedzie się z powodu błędu (MQRC_RETAINED_MSG_Q_ERROR lub MQRC_RETAINED_NOT_DOSTARCZONY), tworzenie subskrypcji nie powiedzie się.

ŻĄDANIE MQSO_PUBLICATIONS_ON_REQUEST

Ustawienie tej opcji oznacza, że subskrybent będzie żądał informacji w szczególności, gdy jest to wymagane. Menedżer kolejek nie wysyła niezamówionych komunikatów do subskrybenta. Zachowana publikacja (lub ewentualnie wiele publikacji, jeśli w temacie podano znak wieloznaczny) jest przesyłana do subskrybenta za każdym razem, gdy wywołanie MQSUBRQ jest wykonywane przy użyciu uchwytu Hsub z poprzedniego wywołania MQSUB. Żadne publikacje nie są wysyłane w wyniku wywołania MQSUB za pomocą tej opcji. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME ta opcja jest ustawiana, jeśli jest odpowiednia.

Ta opcja nie jest poprawna w połączeniu z SubLevel większym niż 1.

Opcje odczytu z wyprzedzeniem: Następujące opcje kontrolują, czy komunikaty nietrwałe są wysyłane do aplikacji z wyprzedzeniem o żądającej ich aplikacji.

MQSO_READ_AHEAD_AS_Q_DEF

Jeśli wywołanie MQSUB korzysta z uchwytu zarządzanego, domyślny atrybut odczytu z wyprzedzeniem kolejki modelowej powiązanej z tematem subskrybowanym w celu określenia, czy komunikaty są wysyłane do aplikacji, zanim aplikacja je zażąda.

Jest to wartość domyślna.

MQSO_NO_READ_AHEAD

Jeśli wywołanie MQSUB korzysta z zarządzanego uchwytu, komunikaty nie są wysyłane do aplikacji, zanim aplikacja je zażąda.

MQSO_READ_AHEAD

Jeśli wywołanie MQSUB korzysta z zarządzanego uchwytu, komunikaty mogą zostać wysłane do aplikacji, zanim aplikacja je zażąda.

Uwaga:

Do opcji odczytu z wyprzedzeniem mają zastosowanie następujące uwagi:

1. Można określić tylko jedną z tych opcji. Jeśli określono zarówno MQOO_READ_AHEAD, jak i MQOO_NO_READ_AHEAD, zwracany jest kod przyczyny MQRC_OPTIONS_ERROR. Te opcje mają zastosowanie tylko wtedy, gdy określono parametr MQSO_MANAGED.
2. Nie mają one zastosowania w przypadku zmaterializowanych tabel MQSUB, gdy kolejka jest przekazywana, która została wcześniej otwarta. Funkcja odczytu z wyprzedzeniem może nie być włączona w przypadku żądania. Opcje MQGET użyte w pierwszym wywołaniu MQGET mogą uniemożliwić włączenie odczytu z wyprzedzeniem. Ponadto funkcja odczytu z wyprzedzeniem jest wyłączona, gdy klient łączy się z menedżerem kolejek, w którym odczyt z wyprzedzeniem nie jest obsługiwany. Jeśli aplikacja nie jest uruchomiona jako klient IBM MQ, opcje te są ignorowane.

Opcje wieloznaczne: Następujące opcje sterują sposobem interpretowania znaków wieloznacznych w łańcuchu udostępnionym w polu ObjectString w tabeli MQSD. Można określić tylko jedną z tych opcji. W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER te opcje wieloznaczne nie mogą być zmieniane. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME, ustawiana jest odpowiednia opcja znaków wieloznacznych.

MQSO_WILDCARD_CHAR

Znaki wieloznaczne działają tylko na znakach w łańcuchu tematu.

Zachowanie zdefiniowane przez parametr MQSO_WILDCARD_CHAR jest przedstawione w poniższej tabeli.

Znak specjalny	Zachowanie
Prawy ukośnik (/)	Bez znaczenia, tylko inny znak
Gwiazdka (*)	Znak wieloznaczny, zero lub więcej znaków
Znak zapytania (?)	Znak wieloznaczny, 1 znak
Znak procentu (%)	Znak zmiany znaczenia, który umożliwia użycie znaków (*), (?) lub (%) w łańcuchu i nie może być interpretowane jako znak specjalny, na przykład (% *), (%?) lub (%%).

Na przykład opublikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

pasuje do subskrybentów korzystających z następujących tematów:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?e12/level3/level4
```

Uwaga: Użycie znaków wieloznacznych zapewnia dokładnie znaczenie podane w systemach IBM MQ V6 i WebSphere MB V6 , gdy używane są komunikaty w formacie MQRFH1 w celu publikowania/ subskrybowania. Zaleca się, aby nie było to używane w przypadku nowo napisanych aplikacji i jest używane tylko w przypadku aplikacji, które wcześniej były uruchamiane względem tej wersji i nie zostały zmienione w celu użycia domyślnego zachowania ze znakami wieloznacznymi zgodnie z opisem w sekcji MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TOPIC

Znaki wieloznaczne działają tylko na elementach tematu w łańcuchu tematu. Jest to zachowanie domyślne, jeśli nie zostanie wybrane żadne działanie.

Zachowanie wymagane przez parametr MQSO_WILDCARD_TOPIC jest przedstawione w poniższej tabeli:

Znak specjalny	Zachowanie
(/)	Separator poziomu tematu
Znak liczby (#)	Znak wieloznaczny: wiele poziomów tematów
Znak plusa (+)	Znak wieloznaczny: pojedynczy poziom tematu

Uwagi:

Znaki (+) i (#) nie są traktowane jako znaki wieloznaczne, jeśli są one mieszane z innymi znakami (w tym samymi znakami) na poziomie tematu. W poniższym łańcuchu znaki (#) i (+) są traktowane jak zwykłe znaki.

```
level0/level1/#+/level3/level#
```

Na przykład opublikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

pasuje do subskrybentów korzystających z następujących tematów:

```
#  
/#  
/ level0/level1/level2/level3/#  
/ level0/level1/+/level3/level4
```

Inne opcje: Następujące opcje sterują sposobem wydania wywołania API, a nie subskrypcją. W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME opcje te nie ulegają zmianie. Więcej szczegółów na ten temat zawiera sekcja [“Identyfikator AlternateUser\(MQCHAR12\)”](#) na stronie 569.

MQSO_ALTERNATE_USER_AUTHORITY

Pole Identyfikator AlternateUser (Identyfikator użytkownika) zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności wywołania MQSUB. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy ten identyfikator użytkownika AlternateUser ma uprawnienia do otwarcia obiektu z określonymi opcjami dostępu, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma do tego uprawnienia.

MQSO_SET_CORREL_ID

Subskrypcja jest używana do używania identyfikatora korelacji podanego w polu *SubCorrelId*. Jeśli ta opcja nie zostanie podana, identyfikator korelacji jest automatycznie tworzony przez menedżer kolejek w czasie subskrypcji i jest zwracany do aplikacji w polu *SubCorrelId*. Więcej informacji na ten temat zawiera sekcja [“Identyfikator SubCorrel\(MQBYTE24\)”](#) na stronie 582.

Tej opcji nie można łączyć z opcją MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT

Subskrypcja jest używana do używania znaczników rozliczeniowych i danych tożsamości aplikacji podanych w polach *PubAccountingToken* i *PubApplIdentityData*.

Jeśli ta opcja jest określona, to ta sama kontrola autoryzacji jest przeprowadzana tak, jakby kolejka docelowa była dostępna za pomocą wywołania MQOPEN z opcją MQOO_SET_IDENTITY_CONTEXT, z wyjątkiem sytuacji, w której używana jest również opcja MQSO_MANAGED, w której to przypadku nie ma uprawnień do sprawdzania autoryzacji w kolejce docelowej.

Jeśli ta opcja nie zostanie podana, publikacje wysłane do tego subskrybenta mają domyślnie powiązane z nimi informacje o kontekście:

Pole w strukturze MQMD	Użyta wartość
<i>UserIdentifier</i>	Identyfikator użytkownika powiązany z subskrypcją w momencie, w którym została wykonana subskrypcja.
<i>AccountingToken</i>	Określana na podstawie środowiska, jeśli jest to możliwe; jeśli nie, należy ustawić wartość MQACT_NONE.
<i>ApplIdentityData</i>	Ustaw na puste

Ta opcja jest poprawna tylko z opcją MQSO_CREATE i MQSO_ALTER. W przypadku użycia z opcją MQSO_RESUME pola *PubAccountingToken* i *PubApplIdentityData* są ignorowane, więc ta opcja nie ma żadnego efektu.

Jeśli subskrypcja została zmieniona bez użycia tej opcji, w której poprzednio subskrypcja dostarczyła informacje o kontekście tożsamości, dla zmienionej subskrypcji generowane są domyślne informacje o kontekście.

Jeśli subskrypcja zezwalająca na użycie różnych identyfikatorów użytkowników przy użyciu opcji MQSO_ANY_USERID jest wznawiana przez inny identyfikator użytkownika, domyślny kontekst tożsamości jest generowany dla nowego identyfikatora użytkownika będącego właścicielem subskrypcji, a wszystkie kolejne publikacje są dostarczane z nowym kontekstem tożsamości.

MQSO_FAIL_IF QUIESCING

Wywołanie MQSUB nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania. W systemie z/OS dla aplikacji CICS lub IMS ta opcja również wymusza, że wywołanie MQSUB nie powiedzie się, jeśli połączenie jest w stanie wygaszania.

Znacznik PubAccounting(MQBYTE32)

Jest to wartość, która będzie znajdować się w polu *AccountingToken* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *AccountingToken* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu. Więcej informacji na temat pola *AccountingToken* w strukturze MQMD zawiera sekcja [“AccountingToken \(MQBYTE32\)” na stronie 420](#)

Dla pola *PubAccountingToken* można użyć następującej wartości specjalnej:

MQACT_NONE

Nie określono znacznika rozliczeniowego.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQACT_NONE_ARRAY; ma ona taką samą wartość jak MQACT_NONE, ale jest tablicą znaków zamiast łańcucha.

Jeśli opcja MQSO_SET_IDENTITY_CONTEXT nie jest określona, znacznik rozliczeniowy jest generowany przez menedżer kolejek jako domyślne informacje kontekstowe, a to pole jest polem wyjściowym zawierającym *AccountingToken*, które zostanie ustawione w każdym komunikacie opublikowanym dla tej subskrypcji.

Jeśli zostanie podana opcja MQSO_SET_IDENTITY_CONTEXT, znacznik rozliczeniowy jest generowany przez użytkownika, a pole to jest polem wejściowym zawierającym *AccountingToken*, który ma zostać ustawiony w każdej publikacji dla tej subskrypcji.

Długość tego pola jest podana przez wartość MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową tego pola jest MQACT_NONE.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER można zmienić wartość parametru *AccountingToken* w dowolnych przyszłych komunikatach publikowania.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole jest ustawione na bieżącą wartość *AccountingToken* używaną dla subskrypcji.

PubApplIdentityData (MQCHAR32)

Jest to wartość, która znajduje się w polu *ApplIdentityData* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *ApplIdentityData* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu. Więcej informacji na temat pola *ApplIdentityData* w strukturze MQMD zawiera sekcja [“Dane ApplIdentity\(MQCHAR32\)” na stronie 422](#)

Jeśli opcja MQSO_SET_IDENTITY_CONTEXT nie jest określona, wartość *ApplIdentityData*, która jest ustawiona w każdym komunikacie opublikowanym dla tej subskrypcji, jest pusta, jako domyślne informacje o kontekście.

Jeśli zostanie podana opcja MQSO_SET_IDENTITY_CONTEXT, *PubApplIdentityData* jest generowana przez użytkownika, a to pole jest polem wejściowym, które zawiera *ApplIdentityData*, które mają być ustawione w każdej publikacji dla tej subskrypcji.

Długość tego pola jest podana przez wartość MQ_APPL_IDENTITY_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 32 puste znaki w innych językach programowania.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, można zmienić *ApplIdentityData* wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole jest ustawione na bieżącą wartość *ApplIdentityData* używaną dla subskrypcji.

PubPriority (MQLONG)

Jest to wartość, która będzie znajdować się w polu *Priority* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Więcej informacji na temat pola *Priority* w strukturze MQMD zawiera sekcja “Priorytet (MQLONG)” na stronie 448.

Wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następujących wartości specjalnych:

MQPRI_PRIORITY_AS_Q_DEF

Jeśli kolejka subskrypcji jest udostępniana w polu *Hobj* w wywołaniu MQSUB i nie jest to uchwyt zarządzany, priorytet dla komunikatu jest przyjmowany z atrybutu *DefPriority* tej kolejki. Jeśli kolejka jest kolejką klastra lub istnieje więcej niż jedna definicja w ścieżce rozstrzygania nazw kolejek, priorytet jest określany, gdy komunikat publikacji jest umieszczany w kolejce w sposób opisany w sekcji “Priorytet (MQLONG)” na stronie 448.

Jeśli wywołanie MQSUB korzysta z uchwytu zarządzanego, priorytet dla komunikatu jest przyjmowany z atrybutu *DefPriority* kolejki modelowej powiązanej z subskrybowanym tematem.

MQPRI_PRIORITY_AS_PUBLISHED

Priorytet dla wiadomości jest priorytetem pierwotnej publikacji. Jest to początkowa wartość pola.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, można zmienić *Priority* wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME, to pole jest ustawione na bieżący priorytet używany w subskrypcji.

Łańcuch ResObject(MQCHARV)

Jest to długa nazwa obiektu po przetłumaczonej nazwie menedżera kolejek w produkcie *ObjectName*.

Jeśli długa nazwa obiektu jest dostępna w produkcie *ObjectString*, a w produkcie *ObjectName* jest udostępniana żadna wartość, to wartość zwrócona w tym polu jest taka sama, jak podana w składce *ObjectString*.

Jeśli to pole zostanie pominięte (czyli *ResObjectString.VSBufSize* ma wartość zero), to pole *ResObjectString* nie zostanie zwrócone, ale długość jest zwracana w obiekcie *ResObjectString.VSLength*. Jeśli długość jest krótsza niż pełny łańcuch *ResObject*, to jest ona obcinana i zwraca tyle znaków z prawej strony, ile może zmieścić się w podanej długości.

Jeśli wartość *ResObjectString* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RES_OBJECT_STRING_ERROR.

SelectionString (MQCHARV)

Jest to łańcuch używany do udostępniania kryteriów wyboru używanych podczas subskrybowania komunikatów z tematu.

To pole o zmiennej długości zostanie zwrócone w wyniku wywołania MQSUB za pomocą opcji MQSO_RESUME, jeśli bufor został udostępniony, a ponadto w polu VSBufSize istnieje dodatnia długość buforu. Jeśli w wywołaniu nie zostanie podany żaden bufor, tylko długość łańcucha wyboru zostanie zwrócona w polu VSLength w tabeli MQCHARV. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko bajty VSBufSize.

Jeśli wartość *SelectionString* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury „MQCHARV-łańcuch o zmiennej długości” na stronie 298 lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SELECTION_STRING_ERROR.

Użycie opcji *SelectionString* jest opisane w sekcji [Selektory](#).

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQSD_STRUC_ID

Identyfikator struktury deskryptora subskrypcji.

Dla języka programowania C zdefiniowana jest również stała MQSD_STRUC_ID_ARRAY; ma taką samą wartość jak MQSD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQSD_STRUC_ID.

Identyfikator SubCorrel(MQBYTE24)

To pole zawiera identyfikator korelacji wspólny dla wszystkich publikacji zgodnych z tą subskrypcją.



Ostrzeżenie: Identyfikator korelacji może być przekazywany tylko między menedżerami kolejek w klastrze publikowania/subskrypcji, a nie w hierarchii.

Wszystkie publikacje wysłane w celu dopasowania do tej subskrypcji zawierają ten identyfikator korelacji w deskrypcorze komunikatu. Jeśli wiele subskrypcji pobiera swoje publikacje z tej samej kolejki, użycie identyfikatora MQGET według identyfikatora korelacji umożliwia uzyskanie tylko tych publikacji, które mają zostać uzyskane. Ten identyfikator korelacji może być wygenerowany przez menedżera kolejek lub przez użytkownika.

Jeśli opcja MQSO_SET_CORREL_ID nie jest określona, identyfikator korelacji jest generowany przez menedżera kolejek, a to pole jest polem wyjściowym zawierającym identyfikator korelacji, który zostanie ustawiony w każdym komunikacie opublikowanym dla tej subskrypcji. Wygenerowany identyfikator korelacji składa się z 4-bajtowego identyfikatora produktu (AMQX lub CSQM w kodzie ASCII lub EBCDIC), po którym następuje implementacja specyficzna dla produktu w unikalnym łańcuchu.

Jeśli zostanie podana opcja MQSO_SET_CORREL_ID, identyfikator korelacji jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym identyfikator korelacji, który ma być ustawiony w każdej publikacji dla tej subskrypcji. W tym przypadku, jeśli pole zawiera wartość MQCI_NONE, identyfikatorem korelacji ustawionym w każdym komunikacie opublikowanym dla tej subskrypcji jest identyfikator korelacji utworzony przez oryginalny element wstawiony komunikatu.

Jeśli określona jest opcja MQSO_GROUP_SUB, a określony identyfikator korelacji jest taki sam, jak istniejąca grupowa subskrypcja za pomocą tej samej kolejki i nakładającego się łańcucha tematu, tylko najbardziej znacząca subskrypcja w grupie jest udostępniana z kopią publikacji.

Długość tego pola jest podana przez wartość MQ_CORREL_ID_LENGTH. Wartością początkową tego pola jest MQCI_NONE.

Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji MQSO_ALTER, a to pole jest polem wejściowym, to można zmienić identyfikator korelacji subskrypcji, chyba że subskrypcja jest subskrypcją grupową, to znaczy została utworzona przy użyciu opcji MQSO_GROUP_SUB, w którym to przypadku nie można zmienić identyfikatora korelacji subskrypcji.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole jest ustawiane na bieżący identyfikator korelacji dla subskrypcji.

SubExpiry (MQLONG)

Jest to czas wyrażony w dziesiątych częściach sekundy, po upływie których subskrypcja utraci ważność. Po upływie tego okresu nie będą one zgodne z tą subskrypcją. Gdy subskrypcja utraci ważność, publikacje nie są już wysyłane do kolejki. Jednak publikacje, które już tam są, nie są w żaden sposób dotknięte. *SubExpiry* nie ma wpływu na wygaśnięcie publikacji.

Rozpoznawana jest następująca wartość specjalna:

MQEI_UNLIMITED

Subskrypcja ma nieograniczony czas utraty ważności.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, należy zmienić jej ważność.

W przypadku powrotu z wywołania MQSUB za pomocą opcji MQSO_RESUME to pole jest ustawione na pierwotne wygaśnięcie subskrypcji, a nie pozostały czas utraty ważności.

SubLevel (MQLONG)

Jest to poziom powiązany z subskrypcją. Publikacje są dostarczane do tej subskrypcji tylko wtedy, gdy znajdują się w zestawie subskrypcji o najwyższej wartości SubLevel mniejszej lub równej PubLevel używanej w czasie publikacji. Jeśli jednak publikacja została zachowana, nie jest ona już dostępna dla subskrybentów na wyższych poziomach, ponieważ jest ponownie publikowana na poziomie PubLevel 1.

Wartość musi należeć do zakresu od zera do 9. Zero jest najniższym poziomem.

Wartością początkową tego pola jest 1.

Więcej informacji na ten temat zawiera sekcja [Intercepting publications](#).

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER nie można zmienić wartości parametru SubLevel.

Łączenie elementu SubLevel z wartością większą niż 1 z opcją MQSO_PUBLICATIONS_ON_REQUEST nie jest dozwolone.

W przypadku powrotu z wywołania MQSUB za pomocą komendy MQSO_RESUME to pole jest ustawiane na bieżący poziom używany w subskrypcji.

Dane SubUser(MQCHARV)

Określa dane użytkownika subskrypcji. Dane podane w subskrypcji w tym polu zostaną dołączone jako właściwość komunikatu danych MQSubUserw każdej publikacji wysłanej do tej subskrypcji.

Maksymalna długość *SubUserData* wynosi 10240.

Jeśli wartość *SubUserData* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury [MQCHARV](#) lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SUB_USER_DATA_ERROR.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQCHARV.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER, dane użytkownika subskrypcji mogą zostać zmienione.

To pole o zmiennej długości jest zwracane w wyniku wywołania MQSUB za pomocą opcji MQSO_RESUME, jeśli bufor jest udostępniony, a w programie *VSBufLen* istnieje dodatnia długość buforu. Jeśli w wywołaniu nie zostanie podany żaden bufor, w polu *VSLength* obiektu MQCHARV zwracana jest tylko długość danych użytkownika subskrypcji. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko *VSBufLen* bajtów.

SubName (MQCHARV)

Określa nazwę subskrypcji. To pole jest wymagane tylko wtedy, gdy parametr *Options* określa opcję MQSO_DURABLE, ale jeśli zostanie ona podana, będzie również używana przez menedżera kolejek dla MQSO_NON_DURABLE.

Jeśli zostanie podana, wartość *SubName* musi być unikalna w obrębie menedżera kolejek, ponieważ jest to metoda używana do identyfikowania subskrypcji.

Maksymalna długość *SubName* wynosi 10240.

To pole służy dwóm celom. W przypadku subskrypcji MQSO_DURABLE to pole służy do identyfikowania subskrypcji, dzięki czemu można ją wznowić po utworzeniu subskrypcji, jeśli użytkownik zamknął uchwyt w subskrypcji (za pomocą opcji MQCO_KEEP_SUB) lub został odłączony od menedżera kolejek. W tym celu należy użyć wywołania MQSUB z opcją MQSO_RESUME. Jest ona również wyświetlana w widoku administracyjnym subskrypcji w polu SUBNAME w polu DISPLAY SBSTATUS.

Jeśli produkt *SubName* został podany niepoprawnie, to zgodnie z opisem sposobu użycia struktury MQCHARV jest pozostawiony, gdy jest wymagany (to znaczy *SubName.VSLength* jest zerem), lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SUB_NAME_ERROR.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQCHARV.

Jeśli istniejąca subskrypcja zostanie zmieniona za pomocą opcji MQSO_ALTER, nie można zmienić nazwy subskrypcji, ponieważ jest to pole identyfikujące używane do znalezienia przywoływanej subskrypcji. Nie jest on zmieniany na wyjściu z wywołania MQSUB z opcją MQSO_RESUME.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQSD_VERSION_1

Struktura deskryptora subskrypcji Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQSD_CURRENT_VERSION

Bieżąca wersja struktury deskryptora subskrypcji.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQSD_VERSION_1.

Korzystanie z łańcuchów tematów

Temat jest konstruowany z podtematu zidentyfikowanego w obiekcie tematu oraz podtematu udostępnionego przez aplikację. Można użyć podtematu jako nazwy tematu lub połączyć je w celu utworzenia nowej nazwy tematu.

W programie MQI pełna nazwa tematu jest tworzona przez program MQOPEN. Składa się on z dwóch pól używanych w wywołaniach MQI publikowania/subskrybowania, w podanej kolejności:

1. Atrybut **TOPICSTR** obiektu tematu o nazwie podanej w polu **ObjectName** .
2. Parametr **ObjectString** definiujący podtemat udostępniany przez aplikację.

Wynikowy łańcuch tematu jest zwracany w parametrze **ResObjectString** .

Te pola są uważane za obecne, jeśli pierwszy znak każdego pola nie jest znakiem pustym lub pustym, a długość pola jest większa od zera. Jeśli istnieje tylko jedno z tych pól, jest ono używane bez zmian jako nazwa tematu. Jeśli żadna z tych pól nie ma wartości, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_UNKNOWN_OBJECT_NAME lub MQRC_TOPIC_STRING_ERROR, jeśli pełna nazwa tematu nie jest poprawna.

Jeśli oba pola są obecne, a " / ' jest wstawiany między dwoma elementami wynikowej połączonej nazwy tematu.

Tabela 87 na stronie 585 przedstawia przykłady konkatencji łańcuchów tematów:

Tabela 87. Przykłady konkatencji łańcuchów tematów			
TOPICSTR	ObjectString	Pełna nazwa tematu	Komentarz
Piłka nożna/Szkocja	' '	Piłka nożna/Szkocja	Funkcja TOPICSTR jest używana samodzielnie.
' '	Piłka nożna/Szkocja	Piłka nożna/Szkocja	Obiekt ObjectString jest używany samodzielnie.
Piłka nożna	Oceny	Piłka nożna/Szkocja	Znak ' / ' jest dodawany w punkcie konkatencji.
Piłka nożna	/Szkocja	Piłka nożna//Szkocja	Miedzy dwoma łańcuchami jest tworzony 'pusty węzeł'
Piłka nożna	Oceny	/Football/Scores	Temat rozpoczyna się od 'pustego węzła'

" / ' Znak jest traktowany jako znak specjalny, udostępniając strukturę do pełnej nazwy tematu w obszarze Drzewa tematów nie może być używany z innych powodów, dla których ma wpływ struktura drzewa tematów. Temat " /Football " nie jest taki sam jak temat " Football ".

Następujące znaki wieloznaczne są znakami specjalnymi:

- plus znak " + '
- znak liczby " # '
- gwiazdka " * '
- znak zapytania " ? '

Te znaki nie są uznawane za niepoprawne, jednak należy upewnić się, że są one używane. Podczas publikowania mogą nie być używane te znaki w łańcuchach tematów. Publikowanie w łańcuchu tematu z ' # ' lub " + ' mieszany z innymi znakami (włącznie z nimi) na poziomie tematu, można zasubskrybować albo ze schematem zastępczym. Publikowanie w łańcuchu tematu z ' # ' lub " + ' jako jedyny znak między dwoma " / ' Powoduje utworzenie łańcucha tematu, który nie może zostać zasubskrybowany jawnie przez aplikację przy użyciu schematu wieloznacznego MQSO_WILDCARD_TOPIC. Ta sytuacja powoduje, że aplikacja pobierze więcej publikacji niż oczekiwano.

Przykładowy fragment kodu

Ten fragment kodu, wyodrębniony z przykładowego programu Przykład 2: publikator do tematu zmiennej, łączy obiekt tematu ze zmiennym łańcuchem tematu.

```
MQOD      td = {MQOD_DEFAULT}; /* Object Descriptor      */
td.ObjectType = MQOT_TOPIC; /* Object is a topic */
td.Version = MQOD_VERSION_4; /* Descriptor needs to be V4 */
strcpy(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);
```

Wartości początkowe i deklaracje języków dla MQSD

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQSD_STRUC_ID	'SD--'
Version	MQSD_VERSION_1	1

Nazwa pola	Nazwa stałej	Wartość stałej
<i>Options</i>	MQSO_NON_DURABLE,	0
<i>ObjectName</i>	Brak	Pusty łańcuch lub odstępy
<i>AlternateUserId</i>	Brak	Pusty łańcuch lub odstępy
<i>AlternateSecurityId</i>	MQSID_NONE	Wartości null
<i>SubExpiry</i>	MQEI_UNLIMITED	-1
<i>ObjectString</i>	Brak	Nazwy i wartości zdefiniowane dla tabeli MQCHARV
<i>SubName</i>	Brak	Nazwy i wartości zdefiniowane dla tabeli MQCHARV
<i>SubUserData</i>	Brak	Nazwy i wartości zdefiniowane dla tabeli MQCHARV
<i>SubCorrelId</i>	MQCI_NONE	Wartości null
<i>PubPriority</i>	MQPRI_PRIORITY_AS_Q_DEF	-3
<i>PubAccountingToken</i>	MQACT_NONE	Wartości null
<i>PubApplIdentityData</i>	Brak	Pusty łańcuch lub odstępy
<i>SelectionString</i>	Brak	Nazwy i wartości zdefiniowane dla tabeli MQCHARV
<i>SubLevel</i>	Brak	1
<i>ResObjectString</i>	Brak	Nazwy i wartości zdefiniowane dla tabeli MQCHARV

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraParametr MQSD_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQSD MySD = {MQSD_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
    MQCHAR4    StructId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     Options;           /* Options associated with subscribing */
    MQCHAR48   ObjectName;        /* Object name */
    MQCHAR12   AlternateUserId;    /* Alternate user identifier */
    MQBYTE40   AlternateSecurityId; /* Alternate security identifier */
    MQLONG     SubExpiry;         /* Expiry of Subscription */
    MQCHARV    ObjectString;      /* Object Long name */
    MQCHARV    SubName;          /* Subscription name */
    MQCHARV    SubUserData;       /* Subscription User data */
    MQBYTE24   SubCorrelId;       /* Correlation Id related to this subscription */
    MQLONG     PubPriority;        /* Priority set in publications */
    MQBYTE32   PubAccountingToken; /* Accounting Token set in publications */
    MQCHAR32   PubApplIdentityData; /* Appl Identity Data set in publications */
    MQCHARV    SelectionString;   /* Message selector structure */
}
```

```

    MQLONG      SubLevel;          /* Subscription level */
    MQCHARV     ResObjectString;    /* Resolved Long object name*/
    /* Ver:1 */
};

```

Deklaracja języka COBOL

```

** Address of variable length string
20 MQSD-OBJECTSTRING-VSPTR      POINTER.
** Offset of variable length string
20 MQSD-OBJECTSTRING-VSOFFSET   PIC S9(9) BINARY.
** size of buffer
20 MQSD-OBJECTSTRING-VSBUFSIZE   PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-OBJECTSTRING-VSLENGTH   PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-OBJECTSTRING-VSCCSID    PIC S9(9) BINARY.
** Subscription name
15 MQSD-SUBNAME.
** Address of variable length string
20 MQSD-SUBNAME-VSPTR          POINTER.
** Offset of variable length string
20 MQSD-SUBNAME-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
20 MQSD-SUBNAME-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SUBNAME-VSLENGTH      PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SUBNAME-VSCCSID       PIC S9(9) BINARY.
** Subscription User data
15 MQSD-SUBUSERDATA.
** Address of variable length string
20 MQSD-SUBUSERDATA-VSPTR     POINTER.
** Offset of variable length string
20 MQSD-SUBUSERDATA-VSOFFSET   PIC S9(9) BINARY.
** size of buffer
20 MQSD-SUBUSERDATA-VSBUFSIZE   PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SUBUSERDATA-VSLENGTH   PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SUBUSERDATA-VSCCSID    PIC S9(9) BINARY.
** Correlation Id related to this subscription
15 MQSD-SUBCORRELID          PIC X(24).
** Priority set in publications
15 MQSD-PUBPRIORITY          PIC S9(9) BINARY.
** Accounting Token set in publications
15 MQSD-PUBACCOUNTINGTOKEN    PIC X(32).
** Appl Identity Data set in publications
15 MQSD-PUBAPPLIDENTITYDATA    PIC X(32).
** Message Selector
15 MQSD-SELECTIONSTRING.
** Address of variable length string
20 MQSD-SELECTIONSTRING-VSPTR  POINTER.
** Offset of variable length string
20 MQSD-SELECTIONSTRING-VSOFFSET PIC S9(9) BINARY.
** size of buffer
20 MQSD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSD-SELECTIONSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSD-SELECTIONSTRING-VSCCSID PIC S9(9) BINARY.
** Selection criteria
20 MQSD-SELECTIONSTRING-SUBLEVEL PIC S9(9) BINARY.
** Long object name
20 MQSD-SELECTIONSTRING-RESOBJSTRING PIC S9(9) BINARY.

```

Deklaracja PL/I

```

dcl
1 MQSD based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      fixed bin(31), /* Structure version number */
3 Options      fixed bin(31), /* Options associated with subscribing */
3 ObjectName   char(48),     /* Object name */
3 AlternateUserId char(12),   /* Alternate user identifier */
3 AlternateSecurityId char(40), /* Alternate security identifier */
3 SubExpiry    fixed bin(31), /* Expiry of Subscription */

```

```

3 ObjectString,          /* Object Long name */
5 VSPtr      pointer,    /* Address of variable length string */
5 VSOffset    fixed bin(31), /* Offset of variable length string */
5 VSBufSize   fixed bin(31), /* size of buffer */
5 VSLength    fixed bin(31), /* Length of variable length string */
5 VSCCSID     fixed bin(31); /* CCSID of variable length string */
3 SubName,          /* Subscription name */
5 VSPtr      pointer,    /* Address of variable length string */
5 VSOffset    fixed bin(31), /* Offset of variable length string */
5 VSBufSize   fixed bin(31), /* size of buffer */
5 VSLength    fixed bin(31), /* Length of variable length string */
5 VSCCSID     fixed bin(31); /* CCSID of variable length string */
3 SubUserData,      /* Subscription User data */
5 VSPtr      pointer,    /* Address of variable length string */
5 VSOffset    fixed bin(31), /* Offset of variable length string */
5 VSBufSize   fixed bin(31), /* size of buffer */
5 VSLength    fixed bin(31), /* Length of variable length string */
5 VSCCSID     fixed bin(31), /* CCSID of variable length string */
3 SubCorrelId char(24), /* Correlation Id related to this subscription */
3 PubPriority    fixed bin(31), /* Priority set in publications */
3 PubAccountingToken char(32), /* Accounting Token set in publications */
3 PubApplIdentityData char(32), /* Appl Identity Data set in publications */
3 SelectionString, /* Message Selection */
5 VSPtr      pointer,    /* Address of variable length string */
5 VSOffset    fixed bin(31), /* Offset of variable length string */
5 VSBufSize   fixed bin(31), /* size of buffer */
5 VSLength    fixed bin(31), /* Length of variable length string */
5 VSCCSID     fixed bin(31), /* CCSID of variable length string */
3 SubLevel     fixed bin(31), /* Subscription level */
3 ResObjectString, /* Resolved Long object name */
5 VSPtr      pointer,    /* Address of variable length string */
5 VSOffset    fixed bin(31), /* Offset of variable length string */
5 VSBufSize   fixed bin(31), /* size of buffer */
5 VSLength    fixed bin(31), /* Length of variable length string */
5 VSCCSID     fixed bin(31); /* CCSID of variable length string */

```

Deklaracja High Level Assembler

```

MQSD          DSECT
MQSD_STRUCID   DS   CL4  Structure identifier
MQSD_VERSION   DS   F    Structure version number
MQSD_OPTIONS   DS   F    Options associated with subscribing
MQSD_OBJECTNAME DS   CL48 Object name
MQSD_ALTERNATEUSERID DS   CL12 Alternate user identifier
MQSD_ALTERNATESECURITYID DS   CL40 Alternate security identifier
MQSD_SUBEXPIRY DS   F    Expiry of Subscription
MQSD_OBJECTSTRING DS   OF  Object Long name
MQSD_OBJECTSTRING_VSPTR DS   F    Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET DS   F    Offset of variable length string
MQSD_OBJECTSTRING_VSBUFSIZE DS   F    size of buffer
MQSD_OBJECTSTRING_VSLENGTH DS   F    Length of variable length string
MQSD_OBJECTSTRING_VSCCSID DS   F    CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH EQU *-MQSD_OBJECTSTRING
ORG MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA DS   CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME   DS   OF  Subscription name
MQSD_SUBNAME_VSPTR DS   F    Address of variable length string
MQSD_SUBNAME_VSOFFSET DS   F    Offset of variable length string
MQSD_SUBNAME_VSBUFSIZE DS   F    size of buffer
MQSD_SUBNAME_VSLENGTH DS   F    Length of variable length string
MQSD_SUBNAME_VSCCSID DS   F    CCSID of variable length string
MQSD_SUBNAME_LENGTH EQU *-MQSD_SUBNAME
ORG MQSD_SUBNAME
MQSD_SUBNAME_AREA DS   CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA DS   OF  Subscription User data
MQSD_SUBUSERDATA_VSPTR DS   F    Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET DS   F    Offset of variable length string
MQSD_SUBUSERDATA_VSBUFSIZE DS   F    size of buffer
MQSD_SUBUSERDATA_VSLENGTH DS   F    Length of variable length string
MQSD_SUBUSERDATA_VSCCSID DS   F    CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH EQU *-MQSD_SUBUSERDATA
ORG MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA DS   CL(MQSD_SUBUSERDATA_LENGTH)

```



```

*
MQSD_SUBCORRELID      DS  CL24 Correlation Id related to this subscription
MQSD_PUBPRIORITY      DS  F   Priority set in publications
MQSD_PUBACCOUNTINGTOKEN DS  CL32 Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA DS  CL32 Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING  DS  F   Message Selector
MQSD_SELECTIONSTRING_VSPTR DS  F   Address of variable length string
MQSD_SELECTIONSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_SELECTIONSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID DS  F   CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH EQU *- MQSD_SELECTIONSTRING
ORG MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA DS  CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL          DS  F   Subscription level
*
MQSD_RESOBJECTSTRING   DS  F   Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR DS  F   Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_RESOBJECTSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID DS  F   CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH EQU *- MQSD_RESOBJECTSTRING
ORG MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA DS  CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH            EQU  *-MQSD
ORG MQSD
MQSD_AREA              DS  CL(MQSD_LENGTH)

```

MQSMPO-Ustawianie opcji właściwości komunikatu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 88. Pola w MQSMPO		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje
<i>ValueEncoding</i>	Kodowanie wartości właściwości	ValueEncoding
<i>ValueCCSID</i>	Zestaw znaków wartości właściwości	ValueCCSID

Przegląd produktu MQSMPO

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ .

Cel: Struktura produktu **MQSMPO** umożliwia aplikacjom określanie opcji, które sterują sposobem ustawiania właściwości komunikatów. Struktura jest parametrem wejściowym w wywołaniu **MQSETMP** .

Zestaw znaków i kodowanie: Dane w programie **MQSMPO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (**MQENC_NATIVE**).

Pola dla MQSMPO

Struktura MQSMPO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

Opcje (MQLONG)

Opcje lokalizacji: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości:

MQSMPO_SET_FIRST

Ustawia wartość pierwszej właściwości, która jest zgodna z podaną nazwą (lub jeśli nie istnieje), dodaje nową właściwość po wszystkich innych właściwościach z pasującą hierarchią.

MQSMPO_SET_PROP_UNDER_CURSOR

Ustawia wartość właściwości wskazanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET, lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli wskaźnik właściwości na podstawie kursora właściwości został usunięty, wywołanie nie powiedzie się i kod zakończenia MQCC_FAILED i kod przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR

Ustawia nową właściwość przed właściwością wskazanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET, lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli wskaźnik właściwości na podstawie kursora właściwości został usunięty, wywołanie nie powiedzie się i kod zakończenia MQCC_FAILED i kod przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_AFTER_CURSOR

Ustawia nową właściwość po właściwości wskazanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET, lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli wskaźnik właściwości na podstawie kursora właściwości został usunięty, wywołanie nie powiedzie się i kod zakończenia MQCC_FAILED i kod przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_APPEND_PROPERTY,

Powoduje, że nowa właściwość zostanie dodana po wszystkich innych właściwościach z pasującą hierarchią. Jeśli istnieje co najmniej jedna właściwość, która jest zgodna z podaną nazwą, nowa właściwość zostanie dodana na końcu po zakończeniu tej listy właściwości.

Ta opcja umożliwia utworzenie listy właściwości o tej samej nazwie.

Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

MQSMPO_NONE,

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MQSMPO_SET_FIRST.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQSMPO_STRUC_ID,

Identyfikator struktury opcji właściwości zestawu komunikatów.

Dla języka programowania w języku C jest również zdefiniowana stała **MQSMPO_STRUC_ID_ARRAY**. Ma ona taką samą wartość jak **MQSMPO_STRUC_ID**, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQSMPO_STRUC_ID**.

ValueCCSID (MQLONG)

Zestaw znaków wartości właściwości, który ma zostać ustawiony, jeśli wartość jest łańcuchem znaków.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQCCSI_APPL**.

ValueEncoding (MQLONG)

Kodowanie wartości właściwości, która ma zostać ustawiona, jeśli wartość jest liczbowa.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQENC_NATIVE**.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQSMPO_VERSION_1

Version-1 ustawia strukturę opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

MQSMPO_CURRENT_VERSION

Bieżąca wersja struktury opcji właściwości komunikatu zestawu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **MQSMPO_VERSION_1**.

Wartości początkowe i deklaracje języków dla MQSMPO

Tabela 89. Początkowe wartości pól w MQSMPO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQSMPO_STRUC_ID,	'SMPO'
<i>Version</i>	MQSMPO_VERSION_1	1
<i>Options</i>	MQSMPO_NONE,	0
<i>ValueEncoding</i>	MQENC_NATIVE	Zależy od środowiska
<i>ValueCCSID</i>	MQCCSI_APPL	-3
Uwagi:		
1. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.		
2. W języku programowania C: zmienna makraParametr MQSMPO_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQSMPO MySMPO = {MQSMPO_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQSETMP */
    MQLONG     ValueEncoding;    /* Encoding of Value */
    MQLONG     ValueCCSID;       /* Character set identifier of Value */
};
```

Deklaracja języka COBOL

```
**      MQSMPO structure
10 MQSMPO.
**      Structure identifier
15 MQSMPO-STRUCID      PIC X(4).
**      Structure version number
15 MQSMPO-VERSION     PIC S9(9) BINARY.
**      Options that control the action of MQSETMP
15 MQSMPO-OPTIONS     PIC S9(9) BINARY.
**      Encoding of VALUE
15 MQSMPO-VALUEENCODING PIC S9(9) BINARY.
**      Character set identifier of VALUE
15 MQSMPO-VALUECCSID  PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1 MQSMPO based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),   /* Structure version number */
3 Options      fixed bin(31),   /* Options that control the action of MQSETMP */
3 ValueEncoding fixed bin(31),  /* Encoding of Value */
3 ValueCCSID   fixed bin(31),   /* Character set identifier of Value */
```

Deklaracja High Level Assembler

```
MQSMPO          DSECT
MQSMPO_STRUCID  DS    CL4      Structure identifier
MQSMPO_VERSION  DS    F        Structure version number
MQSMPO_OPTIONS  DS    F        Options that control the action of
*                MQSETMP
MQSMPO_VALUEENCODING DS    F    Encoding of VALUE
MQSMPO_VALUECCSID DS    F      Character set identifier of VALUE
MQSMPO_LENGTH   EQU    *-MQSMPO
MQSMPO_AREA     DS    CL(MQSMPO_LENGTH)
```

MQSRO-opcje żądania subskrypcji

W tej sekcji opisano opcje żądania subskrypcji, jakie pola zawiera oraz wartości początkowe tych pól.

Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>Options</i>	Opcje	Opcje
<i>NumPubs</i>	Liczba publikacji	NumPubs

Przegląd dla MQSRO

Dostępność: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS plus IBM MQ MQI clients connected to these systems.

Cel: Struktura MQSRO umożliwia aplikacji określanie opcji, które sterują sposobem wykonania żądania subskrypcji. Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSUBRQ.

Wersja: Bieżąca wersja obiektu MQSRO to MQSRO_VERSION_1.

Zestaw znaków i kodowanie: Dane w tabeli MQSRO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE. Jeśli jednak aplikacja jest uruchomiona jako klient MQI produktu MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Pola dla MQSRO

Struktura MQSRO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

NumPubs (MQLONG)

Jest to pole wyjściowe, zwrócone do aplikacji w celu wskazania liczby publikacji wysłanych do kolejki subskrypcji w wyniku tego wywołania. Mimo że ta liczba publikacji została wysłana w wyniku tego wywołania, nie ma gwarancji, że ta liczba komunikatów będzie dostępna dla aplikacji do pobrania, zwłaszcza jeśli są to komunikaty nietrwałe.

Jeśli temat zasubskrybował znak wieloznaczny, może istnieć więcej niż jedna publikacja. Jeśli w łańcuchu tematu nie było żadnych znaków wieloznacznych podczas tworzenia subskrypcji reprezentowanej przez *Hsub*, to w wyniku tego wywołania w większości wysyłane jest tylko jedna publikacja.

Opcje (MQLONG)

Należy podać jedną z następujących opcji. Można podać tylko jedną opcję.

MQSRO_FAIL_IF_QUIESCING

Wywołanie MQSUBRQ nie powiodło się, jeśli menedżer kolejek znajduje się w stanie wygaszania. W przypadku produktu z/OS dla aplikacji CICS lub IMS ta opcja również wymusza niepowodzenie wywołania MQSUBRQ, jeśli połączenie jest w stanie wygaszania.

Opcja domyślna: Jeśli opisana wcześniej opcja nie jest wymagana, należy użyć następującej opcji:

MQSRO_NONE

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

MQSRO_NONE pomaga w dokumentacji programu. Chociaż nie jest zamierzone, aby ta opcja była używana z innymi, ponieważ jej wartość wynosi zero, nie można jej wykryć.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQSRO_STRUC_ID,

Identyfikator struktury opcji żądania subskrypcji.

Dla języka programowania C zdefiniowana jest również stała MQSRO_STRUC_ID_ARRAY; ma taką samą wartość jak MQSRO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQSRO_STRUC_ID.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQSRO_VERSION_1

Struktura opcji żądania subskrypcji Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQSRO_CURRENT_VERSION

Bieżąca wersja struktury opcji żądania subskrypcji.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MQSRO_VERSION_1.

Wartości początkowe i deklaracje języków dla MQSRO

Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQSRO_STRUC_ID,	'SRO¬'
<i>Version</i>	MQSRO_VERSION_1	1
<i>Options</i>	MQSRO_NONE	0
<i>NumPubs</i>	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.
2. W języku programowania C: zmienna makraParametr MQSRO_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     Options;          /* Options that control the action of MQSUBRQ */
    MQLONG     NumPubs;          /* Number of publications sent */
    /* Ver:1 */
};
```

Deklaracja języka COBOL

```
** MQSRO structure
10  MQSRO.
** Structure identifier
15  MQSRO-STRUCID          PIC X(4).
** Structure version number
15  MQSRO-VERSION         PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
15  MQSRO-OPTIONS         PIC S9(9) BINARY.
** Number of publications sent
15  MQSRO-NUMPUBS         PIC S9(9) BINARY.
```

Deklaracja PL/I

```
dcl
1  MQSRO based,
3  StrucId      char(4),          /* Structure identifier */
3  Version      fixed bin(31),   /* Structure version number */
3  Options      fixed bin(31),   /* Options that control the action of MQSUBRQ */
3  NumPubs      fixed bin(31);   /* Number of publications sent */
```

```

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-struktura raportowania statusu

W poniższej tabeli podsumowano pola w strukturze.

Tabela 90. Pola w tabeli MQSTS		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>CompCode</i>	Kod zakończenia pierwszego błędu	CompCode
<i>Reason</i>	Kod przyczyny pierwszego błędu	Powód
<i>PutSuccessCount</i>	Liczba pomyślnych asynchronicznych wywołań put	SuccessCount
<i>PutWarningCount</i>	Liczba asynchronicznych wywołań put, które miały ostrzeżenia	WarningCount
<i>PutFailureCount</i>	Liczba niepomyślnych asynchronicznych wywołań put	FailureCount
<i>ObjectType</i>	Typ uszkodzonego obiektu	ObjectType
<i>ObjectName</i>	Nazwa niesprawnego obiektu	ObjectName
<i>ObjectQMgrName</i>	Nazwa menedżera kolejek będącego właścicielem uszkodzonego obiektu	NazwaObjectQMgr
<i>ResolvedObjectName</i>	Rozstrzygnięta nazwa kolejki docelowej	ResolvedObjectNazwa
<i>ResolvedQMgrName</i>	Rozstrzygnięta nazwa docelowego menedżera kolejek	ResolvedQMgrNazwa
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQSTS_VERSION_2.		
<i>ObjectString</i>	Długa nazwa obiektu, w którym wystąpił błąd.	ObjectString
<i>SubName</i>	Nazwa subskrypcji niesprawnej subskrypcji	SubName
<i>OpenOptions</i>	Opcje otwarcia powiązane z niepowodzeniem	OpenOptions
<i>SubOptions</i>	Opcje subskrypcji powiązane z niepowodzeniem	SubOptions

Przegląd produktu MQSTS

Cel: Struktura MQSTS jest parametrem wyjściowym z komendy MQSTAT.

Zestaw znaków i kodowanie: Dane znakowe w tabeli MQSTS znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to nadawane za pomocą atrybutu menedżera kolejek *CodedCharSetId*

menedżera kolejek. Dane liczbowe w tabeli MQSTS znajdują się w rodzimym kodowaniu komputera. Dane te są podawane za pomocą opcji *Kodowanie*.

Użycie: Komenda MQSTAT jest używana do pobierania informacji o statusie. Te informacje są zwracane w strukturze MQSTS. Więcej informacji na temat komendy MQSTAT zawiera sekcja [“MQSTAT-pobieranie informacji o statusie”](#) na stronie 796.

Pola dla MQSTS

Struktura MQSTS zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CompCode (MQLONG)

Kod zakończenia operacji, w której jest raportowana operacja.

Interpretacja wartości CompCode zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Jest to kod zakończenia wynikający z poprzedniej operacji asynchronicznej operacji put dla obiektu określonego w ObjectName.

MQSTAT_TYPE_RECONNECTION

Jeśli połączenie jest ponownie nawiązywane lub nie powiodło się ponowne nawiązanie połączenia, jest to kod zakończenia, który spowodował ponowne nawiązanie połączenia.

Jeśli połączenie jest aktualnie połączone, wartością jest MQCC_OK.

MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne nawiązanie połączenia nie powiodło się, jest to kod zakończenia, który spowodował niepowodzenie ponownego nawiązania połączenia.

Jeśli połączenie jest aktualnie połączone lub ponownie nawiąże połączenie, wartością jest MQCC_OK.

CompCode jest zawsze polem wyjściowym. Jego początkowa wartość to MQCC_OK.

ObjectName (MQCHAR48)

Nazwa zgłaszanego obiektu.

Interpretacja wartości ObjectName zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Jest to nazwa kolejki lub tematu używanego w operacji put, którego niepowodzenie jest raportowane w polach *CompCode* i *Reason* w strukturze MQSTS .

MQSTAT_TYPE_RECONNECTION

Jeśli połączenie jest ponownie nawiązywane, jest to nazwa menedżera kolejek powiązanego z połączeniem.

MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli nawiązanie połączenia nie powiodło się, jest to nazwa obiektu, który spowodował niepowodzenie ponownego połączenia. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze MQSTS .

ObjectName jest polem wyjściowym. Jego początkowa wartość jest łańcuchem pustym w języku C, a 48 pustych znaków w innych językach programowania.

Nazwa ObjectQMGr(MQCHAR48)

Nazwa menedżera kolejek, który jest zgłaszany.

Interpretacja wartości ObjectQMGrName zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Jest to nazwa menedżera kolejek, w którym zdefiniowany jest obiekt *ObjectName* . Nazwa, która jest całkowicie pusta, do pierwszego znaku o wartości NULL lub do końca pola oznacza menedżer kolejek, z którym połączona jest aplikacja (lokalny menedżer kolejek).

MQSTAT_TYPE_RECONNECTION

Puste.

MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli nawiązanie połączenia nie powiodło się, jest to nazwa obiektu, który spowodował niepowodzenie ponownego połączenia. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze MQSTS .

ObjectQMgrName jest polem wyjściowym. Jego wartość jest łańcuchem pustym w języku C, a 48 pustych znaków w innych językach programowania.

ObjectString (MQCHARV)

Długa nazwa obiektu, dla którego zgłaszany jest błąd obiektu. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Interpretacja wartości ObjectString zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Jest to długa nazwa obiektu dla kolejki lub tematu używanego w operacji MQPUT , która nie powiodła się.

MQSTAT_TYPE_RECONNECTION

Łańcuch o zerowej długości

MQSTAT_TYPE_RECONNECTION_ERROR

Jest to długa nazwa obiektu, który spowodował niepowodzenie ponownego nawiązania połączenia.

ObjectString jest polem wyjściowym. Jego początkowa wartość to łańcuch o zerowej długości.

ObjectType (MQLONG)

Typ obiektu o nazwie *ObjectName* , który jest zgłaszany.

Możliwe wartości ObjectType są wymienione w [“MQOT_ * \(typy obiektów i typy obiektów rozszerzonych\)”](#) na stronie 165.

ObjectType jest polem wyjściowym. Jego początkowa wartość to MQOT_Q.

OpenOptions (MQLONG)

OpenOptions używany do otwierania zgłaszanego obiektu. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Wartość OpenOptions zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Zero.

MQSTAT_TYPE_RECONNECTION

Zero.

MQSTAT_TYPE_RECONNECTION_ERROR

OpenOptions , który był używany w przypadku wystąpienia błędu. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze MQSTS .

OpenOptions jest polem wyjściowym. Jego początkowa wartość wynosi zero.

Liczba wywołań PutFailure(MQLONG)

Liczba operacji asynchronicznego put, które nie powiodły się.

Wartość PutFailureCount zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Liczba asynchronicznych operacji put dla obiektu nazwanego w strukturze MQSTS , która została zakończona z programem MQCC_FAILED.

MQSTAT_TYPE_RECONNECTION

Zero.

MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutFailureCount jest polem wyjściowym. Jego początkowa wartość wynosi zero.

Liczba wywołań PutSuccess(MQLONG)

Liczba operacji put asynchronicznych, które powiodły się.

Wartość PutSuccessCount zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Liczba asynchronicznych operacji put dla obiektu nazwanego w strukturze MQSTS , która została zakończona z programem MQCC_OK.

MQSTAT_TYPE_RECONNECTION

Zero.

MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutSuccessCount jest polem wyjściowym. Jego początkowa wartość wynosi zero.

Licznik PutWarning(MQLONG)

Liczba asynchronicznych operacji put, które zakończyły się ostrzeżeniem.

Wartość PutWarningCount zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Liczba asynchronicznych operacji put dla obiektu nazwanego w strukturze MQSTS , która została zakończona z programem MQCC_WARNING.

MQSTAT_TYPE_RECONNECTION

Zero.

MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutWarningCount jest polem wyjściowym. Jego początkowa wartość wynosi zero.

SubName (MQCHARV)

Nazwa niesprawnej subskrypcji. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Interpretacja wartości SubName zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Łańcuch o zerowej długości.

MQSTAT_TYPE_RECONNECTION

Łańcuch o zerowej długości.

MQSTAT_TYPE_RECONNECTION_ERROR

Nazwa subskrypcji, która spowodowała niepowodzenie ponownego nawiązania połączenia. Jeśli żadna nazwa subskrypcji nie jest dostępna lub niepowodzenie nie jest powiązane z subskrypcją, jest to łańcuch o zerowej długości.

SubName jest polem wyjściowym. Jego początkowa wartość to łańcuch o zerowej długości.

SubOptions (MQLONG)

SubOptions używana do otwierania uszkodzonej subskrypcji. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Interpretacja wartości SubOptions zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Zero.

MQSTAT_TYPE_RECONNECTION

Zero.

MQSTAT_TYPE_RECONNECTION_ERROR

SubOptions , który był używany w przypadku wystąpienia błędu. Jeśli niepowodzenie nie jest powiązane z subskrypcją tematu, zwrócona wartość wynosi zero.

SubOptions jest polem wyjściowym. Jego początkowa wartość wynosi zero.

Przyczyna (MQLONG)

Kod przyczyny zgłaszanej operacji.

Interpretacja wartości Reason zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

Jest to kod przyczyny wynikający z poprzedniej operacji asynchronicznej operacji put dla obiektu określonego w ObjectName.

MQSTAT_TYPE_RECONNECTION

Jeśli połączenie jest ponownie nawiązujące połączenie lub nie powiodło się ponowne połączenie, jest to kod przyczyny, który spowodował ponowne nawiązanie ponownego połączenia.

Jeśli połączenie jest aktualnie połączone, wartością jest MQRC_NONE.

MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne nawiązanie połączenia nie powiodło się, jest to kod przyczyny, który spowodował niepowodzenie ponownego nawiązania połączenia.

Jeśli połączenie jest aktualnie połączone lub ponownie nawiąże połączenie, wartością jest MQRC_NONE.

Reason jest polem wyjściowym. Jego początkowa wartość to MQRC_NONE.

Nazwa obiektu ResolvedObject(MQCHAR48)

Nazwa obiektu nazwanego w ObjectName po tłumaczeniu nazwy przez lokalny menedżer kolejek.

Interpretacja wartości ResolvedObjectName zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

ResolvedObjectName to nazwa obiektu nazwanego w ObjectName po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa obiektu, który istnieje w menedżerze kolejek identyfikowany przez produkt ResolvedQMGrName.

MQSTAT_TYPE_RECONNECTION

Puste.

MQSTAT_TYPE_RECONNECTION_ERROR

Puste.

ResolvedObjectName jest polem wyjściowym. Jego początkowa wartość jest łańcuchem pustym w języku C, a 48 pustych znaków w innych językach programowania.

Nazwa ResolvedQMGr(MQCHAR48)

Nazwa docelowego menedżera kolejek po tłumaczeniu nazwy przez lokalny menedżer kolejek.

Interpretacja wartości ResolvedQMGrName zależy od wartości parametru MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR,

ResolvedQMGrName to nazwa docelowego menedżera kolejek po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem obiektu identyfikowanego przez produkt *ResolvedObjectName*. *ResolvedQMGrName* może być nazwą lokalnego menedżera kolejek.

MQSTAT_TYPE_RECONNECTION

Puste.

MQSTAT_TYPE_RECONNECTION_ERROR

Puste.

ResolvedQMGrName jest zawsze polem wyjściowym. Jego początkowa wartość jest łańcuchem pustym w języku C, a 48 pustych znaków w innych językach programowania.

StrucId (MQCHAR4)

Identyfikator struktury raportowania statusu, MQSTS.

StrucId jest identyfikatorem struktury. Wartość musi być następująca:

MQSTS_STRUC_ID

Identyfikator struktury raportowania statusu.

Dla języka programowania w języku C jest również zdefiniowana stała MQSTS_STRUC_ID_ARRAY . Ma ona taką samą wartość jak MQSTS_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucId jest zawsze polem wejściowym. Jego początkowa wartość to MQSTS_STRUC_ID.

Wersja (MQLONG)

Numer wersji struktury.

Wartość musi być albo:

MQSTS_VERSION_1

Struktura raportowania statusu wersji 1.

MQSTS_VERSION_2

Struktura raportowania statusu wersji 2.

Następująca stała określa numer wersji bieżącej wersji:

MQSTS_CURRENT_VERSION

Bieżąca wersja struktury raportowania statusu. Bieżąca wersja to MQSTS_VERSION_2.

Version jest zawsze polem wejściowym. Jego początkowa wartość to MQSTS_VERSION_1.

Wartości początkowe i deklaracje języków dla MQSTS

Tabela 91. Początkowe wartości pól w MQSTS

Nazwa pola	Nazwa stałej	Wartość stałej
<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>	Brak	0
<i>PutWarningCount</i>	Brak	0
<i>PutFailureCount</i>	Brak	0
<i>ObjectType</i>	Kolejka MQOT_Q	1
<i>ObjectName</i>	Brak	Pusty łańcuch lub odstępy
<i>ObjectQMgrName</i>	Brak	Pusty łańcuch lub odstępy
<i>ResolvedObjectName</i>	Brak	Pusty łańcuch lub odstępy
<i>ResolvedQMgrName</i>	Brak	Pusty łańcuch lub odstępy
<i>ObjectString</i>	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<i>SubName</i>	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
<i>OpenOptions</i>	Brak	0
<i>SubOptions</i>	Brak	0

Uwagi:

1. Symbol - reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C zmienna makra MQSTS_DEFAULT zawiera wartości, które są wymienione w tabeli. Można go użyć w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     CompCode;         /* Completion Code of first error */
    MQLONG     Reason;           /* Reason Code of first error */
    MQLONG     PutSuccessCount;   /* Number of Async calls succeeded */
    MQLONG     PutWarningCount;   /* Number of Async calls had warnings */
    MQLONG     PutFailureCount;   /* Number of Async calls had failures */
    MQLONG     ObjectType;        /* Failing object type */
    MQCHAR48    ObjectName;       /* Failing object name */
    MQCHAR48    ObjectQMgrName;   /* Failing object queue manager name */
    MQCHAR48    ResolvedObjectName; /* Resolved name of destination queue */
    MQCHAR48    ResolvedQMgrName; /* Resolved name of destination qmgr */
    /* Ver:1 */
    MQCHARV     ObjectString;     /* Failing object long name */
    MQCHARV     SubName;          /* Failing subscription name */
    MQLONG     OpenOptions;       /* Failing open options */
    MQLONG     SubOptions;        /* Failing subscription options */
}
```

```

/* Ver:2 */
};

```

Deklaracja języka COBOL

```

** MQSTS structure
10 MQSTS.
** Structure identifier
15 MQSTS-STRUCID PIC X(4).
** Structure version number
15 MQSTS-VERSION PIC S9(9) BINARY.
** Completion Code of first error
15 MQSTS-COMPCODE PIC S9(9) BINARY.
** Reason Code of first error
15 MQSTS-REASON PIC S9(9) BINARY.
** Number of Async put calls succeeded
15 MQSTS-PUTSUCCESSCOUNT PIC S9(9) BINARY.
** Number of Async put calls had warnings
15 MQSTS-PUTWARNINGCOUNT PIC S9(9) BINARY.
** Number of Async put calls had failures
15 MQSTS-PUTFAILURECOUNT PIC S9(9) BINARY.
** Failing object type
15 MQSTS-OBJECTTYPE PIC S9(9) BINARY.
** Failing object name
15 MQSTS-OBJECTNAME PIC X(48).
** Failing object queue manager
15 MQSTS-OBJECTQMGRNAME PIC X(48).
** Resolved name of destination queue
15 MQSTS-RESOLVEDOBJECTNAME PIC X(48).
** Resolved name of destination qmgr
15 MQSTS-RESOLVEDQMGRNAME PIC X(48).
** Ver:1 **
** Failing object long name
15 MQSTS-OBJECTSTRING.
** Address of variable length string
20 MQSTS-OBJECTSTRING-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
** Failing subscription name
15 MQSTS-SUBNAME.
** Address of variable length string
20 MQSTS-SUBNAME-VSPTR POINTER.
** Offset of variable length string
20 MQSTS-SUBNAME-VSOFFSET PIC S9(9) BINARY.
** Size of buffer
20 MQSTS-SUBNAME-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
20 MQSTS-SUBNAME-VSLENGTH PIC S9(9) BINARY.
** CCSID of variable length string
20 MQSTS-SUBNAME-VSCCSID PIC S9(9) BINARY.
** Failing open options
15 MQSTS-OPENOPTIONS PIC S9(9) BINARY.
** Failing subscription options
15 MQSTS-SUBOPTIONS PIC S9(9) BINARY.
** Ver:2 **

```

Deklaracja PL/I

```

dcl
1 MQSTS based,
3 StructId          char(4),          /* Structure identifier */
3 Version           fixed bin(31),    /* Structure version number */
3 CompCode          fixed bin(31),    /* Completion code */
3 Reason            fixed bin(31),    /* Reason code */
3 PutSuccessCount   fixed bin(31),    /* Put success count */
3 PutWarningCount   fixed bin(31),    /* Put warning count */
3 PutFailureCount   fixed bin(31),    /* Put failure count */
3 ObjectType        fixed bin(31),    /* Object type */
3 ObjectName        char(48),         /* Object name */
3 ObjectQmgrName    char(48),         /* Object queue manager */

```

```

3 ResolvedObjectName char(48), /* Resolved Object name */
3 ResolvedQmgrName   char(48); /* Resolved Object queue manager */
/* Ver:1 */
3 ObjectString,          /* Failing object long name */
5 VSPtr pointer,         /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31);  /* CCSID of variable length string */
3 SubName,              /* Failing subscription name */
5 VSPtr pointer,         /* Address of variable length string */
5 VSOffset fixed bin(31), /* Offset of variable length string */
5 VSBufSize fixed bin(31), /* Size of buffer */
5 VSLength fixed bin(31), /* Length of variable length string */
5 VSCCSID fixed bin(31);  /* CCSID of variable length string */
3 OpenOptions fixed bin(31), /* Failing open options */
3 SubOptions fixed bin(31); /* Failing subscription options */
/* Ver:2 */

```

Deklaracja High Level Assembler

```

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_VSBUFFSIZE       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_VSBUFFSIZE             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
ORG    MQSTS
MQSTS_AREA                          DS    CL(MQSTS_LENGTH)

```

MQTM-komunikat wyzwacza

W poniższej tabeli podsumowano pola w strukturze.

Tabela 92. Pola w MQTM		
Pole	Opis	Temat
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
QName	Nazwa wyzwanej kolejki	Nazwa QName

Tabela 92. Pola w MQTM (kontynuacja)		
Pole	Opis	Temat
<i>ProcessName</i>	Nazwa obiektu procesu	ProcessName
<i>TriggerData</i>	Dane wyzwalacza	TriggerData
<i>ApplType</i>	Typ aplikacji	ApplType
<i>ApplId</i>	Identyfikator aplikacji	ApplId
<i>EnvData</i>	Dane środowiska	EnvData
<i>UserData</i>	Dane użytkownika	UserData

Przegląd produktu MQTM

Cel: Struktura MQTM opisuje dane w komunikacie wyzwalacza, który jest wysyłany przez menedżer kolejek do aplikacji monitora wyzwalacza, gdy wystąpi zdarzenie wyzwalające dla kolejki.

Ta struktura jest częścią interfejsu programu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska produktu IBM MQ .

Nazwa formatu: MQFMT_TRIGGER.

Zestaw znaków i kodowanie: Dane znakowe w tabeli MQTM znajdują się w zestawie znaków menedżera kolejek, który generuje tabelę MQTM. Dane liczbowe w tabeli MQTM znajdują się w kodowaniu maszyny menedżera kolejek, który generuje program MQTM.

Zestaw znaków i kodowanie tabeli MQTM są podane w polach *CodedCharSetId* i *Encoding* w:

- MQMD (jeśli struktura MQTM znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQTM (wszystkie inne obserwacje).

Użycie: Aplikacja monitorującego wyzwalacza może wymagać przekazania niektórych lub wszystkich informacji w komunikacie wyzwalacza do aplikacji uruchamianej przez aplikację monitora wyzwalacza. Informacje, które mogą być potrzebne w uruchomionej aplikacji, obejmują produkty *QName*, *TriggerData* i *UserData*. Aplikacja wyzwalanie-monitor może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone przez środowisko i wygodne dla uruchomionej aplikacji. Informacje na temat MQTMC2 można znaleźć w sekcji [“MQTMC2 -komunikat wyzwalacza 2 \(format znakowy\)”](#) na stronie 610.

- W systemie z/OS dla aplikacji MQAT_CICS , która jest uruchamiana za pomocą transakcji CKTI, cała struktura komunikatu wyzwalacza MQTM jest dostępna dla uruchomionej transakcji. Informacje te można pobrać za pomocą komendy EXEC CICS RETRIEVE.
- W systemie IBM i aplikacja wyzwalacza-monitor dostarczona z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Więcej informacji na temat używania wyzwalaczy zawiera sekcja [Uruchamianie aplikacji produktu IBM MQ za pomocą wyzwalaczy](#).

MQMD dla komunikatu wyzwalacza: Pola w strukturze MQMD komunikatu wyzwalacza generowanego przez menedżer kolejek są ustawiane w następujący sposób:

Pole w strukturze MQMD	Użyta wartość
<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
<i>Feedback</i>	MQFB_NONE

Pole w strukturze MQMD	Użyta wartość
<i>Encoding</i>	MQENC_NATIVE
<i>CodedCharSetId</i>	Atrybut <i>CodedCharSetId</i> menedżera kolejek
<i>Format</i>	MQFMT_TRIGGER
<i>Priority</i>	Atrybut <i>DefPriority</i> kolejki inicjuj
<i>Persistence</i>	MQPER_NOT_PERSISTENT
<i>MsgId</i>	Unikalna wartość
<i>CorrelId</i>	MQCI_NONE
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Puste
<i>ReplyToQMgr</i>	Nazwa menedżera kolejek.
<i>UserIdentifier</i>	Puste
<i>AccountingToken</i>	MQACT_NONE
<i>ApplIdentityData</i>	Puste
<i>PutApplType</i>	MQAT_QMGR lub, jeśli jest to właściwe dla agenta kanału komunikatów
<i>PutApplName</i>	Pierwsze 28 bajtów nazwy menedżera kolejek
<i>PutDate</i>	Data wysłania komunikatu wyzwalacza
<i>PutTime</i>	Czas wysłania komunikatu wyzwalacza
<i>ApplOriginData</i>	Puste

W celu ustawienia podobnych wartości zaleca się stosowanie aplikacji generującej komunikat wyzwalacza, z wyjątkiem następujących:

- Pole *Priority* można ustawić na wartość MQPRI_PRIORITY_AS_Q_DEF (menedżer kolejek zmieni to ustawienie na priorytet domyślny dla kolejki inicjuj, gdy komunikat jest umieszczany).
- Pole *ReplyToQMgr* można ustawić na puste (menedżer kolejek zmieni to nazwę na nazwę lokalnego menedżera kolejek po umieszczeniu w nim komunikacie).
- Ustaw pola kontekstu jako odpowiednie dla aplikacji.

Pola dla MQTM

Struktura MQTM zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ApplId (MQCHAR256)

Jest to łańcuch znaków identyfikujący aplikację, która ma być uruchomiona, i jest używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *ApplId* obiektu procesu identyfikowanego przez pole *ProcessName*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 881](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

Znaczenie *ApplId* jest określone przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez produkt IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego. Następujące uwagi mają zastosowanie do wskazanych środowisk:

- W systemie z/OSprodukt *ApplId* jest następujący:
 - Identyfikator transakcji CICS, dla aplikacji uruchomionych za pomocą wyzwalacza CICS
 - monitorowanie transakcji CKTI

- Identyfikator transakcji IMS dla aplikacji uruchamianych przy użyciu monitora wyzwalacza IMS CSQQTRMN
- W systemach Windows nazwa programu może być poprzedzona ścieżką napędu i ścieżką do katalogu.
- W systemie IBM inazwa programu może być poprzedzona nazwą biblioteki i/lub znakiem.
- W systemie Systemy UNIXnazwa programu może być poprzedzona ścieżką do katalogu.

Długość tego pola jest podana przez MQ_PROCESS_APPL_ID_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C i 256 znaków odstępu w innych językach programowania.

ApplType (MQLONG)

Identyfikuje on rodzaj programu do uruchomienia i jest używany przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *ApplType* obiektu procesu identyfikowanego przez pole *ProcessName*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów”](#) na stronie 881. Treść tych danych nie ma znaczenia dla menedżera kolejek.

ApplType może mieć jedną z następujących wartości standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu MQAT_USER_FIRST za pomocą MQAT_USER_LAST:

MQAT_AIX

Aplikacja AIX (ta sama wartość jak MQAT_UNIX).

MQAT_BATCH

aplikacja wsadowa

MQAT_BROKER

Aplikacja brokera

MQAT_CICS

CICS.

MQAT_CICS_BRIDGE

Aplikacja CICS bridge.

MQAT_CICS_VSE

CICS/VSE.

MQAT_DOS

Aplikacja IBM MQ MQI client na komputerze PC DOS.

MQAT_IMS

Aplikacja IMS.

MQAT_IMS_BRIDGE

Aplikacja pomostowa IMS.

MQAT_JAVA

Aplikacja Java.

MQAT_MVS

MVS lub aplikacji TSO (taka sama wartość jak MQAT_ZOS).

MQAT_NOTES_AGENT

Lotus Notes Aplikacja agenta.

MQAT_NSK

Aplikacja HP Integrity NonStop Server.

MQAT_OS390

Aplikacja OS/390 (taka sama wartość jak MQAT_ZOS).

MQAT_OS400

Aplikacja IBM i.

MQAT_RRS_BATCH

Aplikacja wsadowa RRS.

MQAT_UNIX

Aplikacja UNIX .

MQAT_UNKNOWN

Aplikacja o nieznanym typie.

UŻYTKOWNIKA_MQAT_

Typ aplikacji zdefiniowany przez użytkownika.

MQAT_VOS

Aplikacja Stratus VOS.

MQAT_WINDOWS

16-bitowa aplikacja Windows .

MQAT_WINDOWS_NT

32-bitowa aplikacja Windows .

MQAT_WLM

Aplikacja menedżera obciążenia produktu z/OS .

MQAT_XCF

XCF.

MQAT_ZOS

Aplikacja z/OS .

MQAT_USER_FIRST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

MQAT_USER_LAST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Wartością początkową tego pola jest 0.

EnvData (MQCHAR128)

Jest to łańcuch znaków zawierający informacje dotyczące środowiska dotyczące aplikacji, która ma być uruchomiona, i jest używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *EnvData* obiektu procesu identyfikowanego przez pole *ProcessName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 881](#) . Treść tych danych nie ma znaczenia dla menedżera kolejek.

W systemie z/OS dla aplikacji CICS uruchomionej przy użyciu transakcji CKTI lub aplikacji IMS , która ma zostać uruchomiona przy użyciu transakcji CSQQTRMN, informacje te nie są używane.

Długość tego pola jest podana przez wartość MQ_PROCESS_ENV_DATA_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 128 znaków odstępu w innych językach programowania.

ProcessName (MQCHAR48)

Jest to nazwa obiektu procesu menedżera kolejek określonego dla kolejki wyzwalanej, która może być używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *ProcessName* kolejki identyfikowanej przez pole *QName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla kolejek” na stronie 845](#).

Nazwy, które są krótsze od zdefiniowanej długości pola, są zawsze dopełniane do prawej strony odstępami; nie są one kończone przedwcześnie znakiem o kodzie zero.

Długość tego pola jest podana przez MQ_PROCESS_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

Nazwa QName (MQCHAR48)

Jest to nazwa kolejki, dla której wystąpiło zdarzenie wyzwalające, i jest używana przez aplikację uruchomioną przez aplikację monitorującego wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *QName* wyzwalanej kolejki. Szczegółowe informacje na temat tego atrybutu zawiera sekcja „Atrybuty dla kolejek” na stronie 845.

Nazwy, które są krótsze od zdefiniowanej długości pola, są dopełniane do prawej strony odstępami; nie są one kończone przedwcześnie znakiem o kodzie zero.

Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQTM_STRUC_ID

Identyfikator struktury komunikatu wyzwalacza.

Dla języka programowania C zdefiniowana jest również stała MQTM_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQTM_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQTM_STRUC_ID.

TriggerData (MQCHAR64)

Jest to dane w formacie wolnoformatowym do użycia przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *TriggerData* kolejki identyfikowanej przez pole *QName*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja „Atrybuty dla kolejek” na stronie 845. Treść tych danych nie ma znaczenia dla menedżera kolejek.

W systemie z/OS dla aplikacji CICS uruchomionej przy użyciu transakcji CKTI informacje te nie są używane.

Długość tego pola jest podana przez parametr MQ_TRIGGER_DATA_LENGTH. Wartością początkową tego pola jest łańcuch o wartości NULL w języku C i 64 znaki odstępu w innych językach programowania.

UserData (MQCHAR128)

Jest to łańcuch znaków zawierający informacje o użytkowniku, które są istotne dla aplikacji, która ma być uruchomiona, i jest używany przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *UserData* obiektu procesu identyfikowanego przez pole *ProcessName*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja „Atrybuty definicji procesów” na stronie 881. Treść tych danych nie ma znaczenia dla menedżera kolejek.

W przypadku systemu Microsoft Windows łańcuch znaków nie może zawierać podwójnych cudzysłówów, jeśli definicja procesu ma być przekazana do produktu **runmqtrm**.

Długość tego pola jest podana przez wartość MQ_PROCESS_USER_DATA_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 128 znaków odstępu w innych językach programowania.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQTM_VERSION_1

Numer wersji struktury komunikatu wyzwalacza.

Następująca stała określa numer wersji bieżącej wersji:

MQTM_CURRENT_VERSION

Bieżąca wersja struktury komunikatu wyzwalacza.

Początkowa wartość tego pola to MQTM_VERSION_1.

Wartości początkowe i deklaracje języka dla produktu MQTM

Tabela 93. Początkowe wartości pól w MQTM dla MQTM

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQTM_STRUC_ID	'TM↵↵'
Version	MQTM_VERSION_1	1
QName	Brak	Pusty łańcuch lub odstępy
ProcessName	Brak	Pusty łańcuch lub odstępy
TriggerData	Brak	Pusty łańcuch lub odstępy
ApplType	Brak	0
ApplId	Brak	Pusty łańcuch lub odstępy
EnvData	Brak	Pusty łańcuch lub odstępy
UserData	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ↵ reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraParametr MQTM_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQLONG     Version;        /* Structure version number */
    MQCHAR48    QName;         /* Name of triggered queue */
    MQCHAR48    ProcessName;   /* Name of process object */
    MQCHAR64    TriggerData;   /* Trigger data */
    MQLONG     ApplType;       /* Application type */
    MQCHAR256   ApplId;        /* Application identifier */
    MQCHAR128   EnvData;       /* Environment data */
    MQCHAR128   UserData;      /* User data */
};
```

Deklaracja języka COBOL

```
**      MQTM structure
10 MQTM.
**      Structure identifier
15 MQTM-STRUCID    PIC X(4).
**      Structure version number
15 MQTM-VERSION    PIC S9(9) BINARY.
**      Name of triggered queue
15 MQTM-QNAME      PIC X(48).
**      Name of process object
15 MQTM-PROCESSNAME PIC X(48).
**      Trigger data
15 MQTM-TRIGGERDATA PIC X(64).
**      Application type
15 MQTM-APPLTYPE    PIC S9(9) BINARY.
**      Application identifier
15 MQTM-APPLID      PIC X(256).
**      Environment data
```

```

15 MQTM-ENVDATA      PIC X(128).
**   User data
15 MQTM-USERDATA     PIC X(128).

```

Deklaracja PL/I

```

dcl
  1 MQTM based,
    3 StrucId      char(4),          /* Structure identifier */
    3 Version      fixed bin(31),    /* Structure version number */
    3 QName        char(48),         /* Name of triggered queue */
    3 ProcessName  char(48),         /* Name of process object */
    3 TriggerData  char(64),         /* Trigger data */
    3 ApplType     fixed bin(31),    /* Application type */
    3 ApplId       char(256),        /* Application identifier */
    3 EnvData      char(128),        /* Environment data */
    3 UserData     char(128);        /* User data */

```

Deklaracja High Level Assembler

```

MQTM          DSECT
MQTM_STRUCID  DS   CL4   Structure identifier
MQTM_VERSION  DS   F     Structure version number
MQTM_QNAME    DS   CL48  Name of triggered queue
MQTM_PROCESSNAME DS CL48  Name of process object
MQTM_TRIGGERDATA DS CL64  Trigger data
MQTM_APPLTYPE DS   F     Application type
MQTM_APPLID   DS   CL256 Application identifier
MQTM_ENVDATA  DS   CL128 Environment data
MQTM_USERDATA DS   CL128 User data
*
MQTM_LENGTH   EQU   *-MQTM
ORG MQTM
MQTM_AREA     DS   CL(MQTM_LENGTH)

```

Wizualna deklaracja podstawowa

```

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 -komunikat wyzwalacza 2 (format znakowy)

W poniższej tabeli podsumowano pola w strukturze.

Tabela 94. Pola w tabeli MQTMC2		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>QName</i>	Nazwa wyzwalanej kolejki	Nazwa QName
<i>ProcessName</i>	Nazwa obiektu procesu	ProcessName
<i>TriggerData</i>	Dane wyzwalacza	TriggerData
<i>ApplType</i>	Typ aplikacji	ApplType

Tabela 94. Pola w tabeli MQTMC2 (kontynuacja)		
Pole	Opis	Temat
<i>ApplId</i>	Identyfikator aplikacji	ApplId
<i>EnvData</i>	Dane środowiska	EnvData
<i>UserData</i>	Dane użytkownika	UserData
<i>QMgrName</i>	Nazwa menedżera kolejek	QMgrName

Przegląd produktu MQTMC2

Cel: Gdy aplikacja wyzwalana przez wyzwalacz pobiera komunikat wyzwalacza (MQTM) z kolejki inicjuj, monitor wyzwalacza może wymagać przekazania niektórych lub wszystkich informacji w komunikacie wyzwalacza do aplikacji, która jest uruchamiana przez monitor wyzwalacza.

Informacje, które mogą być potrzebne w uruchomionej aplikacji, obejmują produkty *QName*, *TriggerData* i *UserData*. Aplikacja monitorującego wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2, w zależności od tego, co jest dozwolone przez środowisko i wygodne dla uruchomionej aplikacji.

Ta struktura jest częścią interfejsu programu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska produktu IBM MQ.

Zestaw znaków i kodowanie: Dane znakowe w tabeli MQTMC2 znajdują się w zestawie znaków lokalnego menedżera kolejek. Dane te są podawane przez atrybut menedżera kolejek produktu *CodedCharSetId*.

Użycie: Struktura MQTMC2 jest bardzo podobna do formatu struktury MQTM. Różnica polega na tym, że pola nieznakowe w MQTM są zmieniane w MQTMC2 na pola znakowe o tej samej długości, a nazwa menedżera kolejek jest dodawana na końcu struktury.

- W systemie z/OS dla aplikacji MQAT_IMS, która jest uruchamiana za pomocą aplikacji CSQQTRMN, struktura MQTMC2 jest dostępna dla uruchomionej aplikacji.
- W systemie IBM i aplikacja monitora wyzwalacza dostarczona z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Pola dla MQTMC2

Struktura MQTMC2 zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ApplId (MQCHAR256)

Identyfikator aplikacji.

Zapoznaj się z polem *ApplId* w strukturze MQTM.

ApplType (MQCHAR4)

Typ aplikacji.

To pole zawsze zawiera spacje, niezależnie od wartości w polu *ApplType* w strukturze MQTM oryginalnego komunikatu wyzwalacza.

EnvData (MQCHAR128)

Dane środowiska.

Zapoznaj się z polem *EnvData* w strukturze MQTM.

ProcessName (MQCHAR48)

Nazwa obiektu procesu.

Zapoznaj się z polem *ProcessName* w strukturze MQTM.

QMgrName (MQCHAR48)
Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, w którym wystąpiło zdarzenie wyzwalające.

Nazwa QName (MQCHAR48)
Nazwa wyzwalanej kolejki.

Zapoznaj się z polem *QName* w strukturze MQTM.

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość musi być następująca:

MQTMC_STRUC_ID

Identyfikator struktury komunikatu wyzwalacza (format znakowy).

Dla języka programowania w języku C jest również zdefiniowana stała MQTMC_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQTMC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

TriggerData (MQCHAR64)
Dane wyzwalacza.

Zapoznaj się z polem *TriggerData* w strukturze MQTM.

UserData (MQCHAR128)
Dane użytkownika.

Zapoznaj się z polem *UserData* w strukturze MQTM.

Wersja (MQCHAR4)
Numer wersji struktury.

Wartość musi być następująca:

MQTMC_VERSION_2

Struktura komunikatu wyzwalacza wersji 2 (format znakowy).

W przypadku języka programowania w języku C jest również zdefiniowana stała MQTMC_VERSION_2_ARRAY. Ma ona taką samą wartość co MQTMC_VERSION_2, ale jest tablicą znaków zamiast łańcucha.

Następująca stała określa numer wersji bieżącej wersji:

MQTMC_CURRENT_VERSION

Bieżąca wersja struktury komunikatu wyzwalacza (format znakowy).

Wartości początkowe i deklaracje języków dla MQTMC2

Tabela 95. Początkowe wartości pól w MQTMC2 dla MQTMC2		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQTMC_STRUC_ID	'TMC'
<i>Version</i>	MQTMC_VERSION_2	'2'
<i>QName</i>	Brak	Pusty łańcuch lub odstęp
<i>ProcessName</i>	Brak	Pusty łańcuch lub odstęp
<i>TriggerData</i>	Brak	Pusty łańcuch lub odstęp

Tabela 95. Początkowe wartości pól w MQTMC2 dla MQTMC2 (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>ApplType</i>	Brak	Puste
<i>ApplId</i>	Brak	Pusty łańcuch lub odstępy
<i>EnvData</i>	Brak	Pusty łańcuch lub odstępy
<i>UserData</i>	Brak	Pusty łańcuch lub odstępy
<i>QMGrName</i>	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraParametr MQTMC2_DEFAULT zawiera wartości wymienione powyżej. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
    MQCHAR4    StrucId;        /* Structure identifier */
    MQCHAR4    Version;        /* Structure version number */
    MQCHAR48    QName;         /* Name of triggered queue */
    MQCHAR48    ProcessName;    /* Name of process object */
    MQCHAR64    TriggerData;    /* Trigger data */
    MQCHAR4    ApplType;       /* Application type */
    MQCHAR256    ApplId;        /* Application identifier */
    MQCHAR128    EnvData;       /* Environment data */
    MQCHAR128    UserData;      /* User data */
    MQCHAR48    QMGrName;      /* Queue manager name */
};
```

Deklaracja języka COBOL

```
** MQTMC2 structure
10 MQTMC2.
** Structure identifier
15 MQTMC2-STRUCID PIC X(4).
** Structure version number
15 MQTMC2-VERSION PIC X(4).
** Name of triggered queue
15 MQTMC2-QNAME PIC X(48).
** Name of process object
15 MQTMC2-PROCESSNAME PIC X(48).
** Trigger data
15 MQTMC2-TRIGGERDATA PIC X(64).
** Application type
15 MQTMC2-APPLTYPE PIC X(4).
** Application identifier
15 MQTMC2-APPLID PIC X(256).
** Environment data
15 MQTMC2-ENVDATA PIC X(128).
** User data
15 MQTMC2-USERSDATA PIC X(128).
** Queue manager name
15 MQTMC2-QMGRNAME PIC X(48).
```

Deklaracja PL/I

```

dcl
1 MQTMC2 based,
3 StrucId      char(4),      /* Structure identifier */
3 Version      char(4),      /* Structure version number */
3 QName        char(48),     /* Name of triggered queue */
3 ProcessName  char(48),     /* Name of process object */
3 TriggerData  char(64),     /* Trigger data */
3 ApplType     char(4),      /* Application type */
3 ApplId       char(256),    /* Application identifier */
3 EnvData      char(128),    /* Environment data */
3 UserData     char(128),    /* User data */
3 QMgrName     char(48);     /* Queue manager name */

```

Deklaracja High Level Assembler

```

MQTMC          DSECT
MQTMC_STRUCID  DS    CL4      Structure identifier
MQTMC_VERSION  DS    CL4      Structure version number
MQTMC_QNAME    DS    CL48     Name of triggered queue
MQTMC_PROCESSNAME DS    CL48  Name of process object
MQTMC_TRIGGERDATA DS    CL64  Trigger data
MQTMC_APPLTYPE DS    CL4      Application type
MQTMC_APPLID   DS    CL256    Application identifier
MQTMC_ENVDATA  DS    CL128    Environment data
MQTMC_USERDATA DS    CL128    User data
MQTMC_QMGRNAME DS    CL48     Queue manager name
*
MQTMC_LENGTH   EQU    *-MQTMC
               ORG    MQTMC
MQTMC_AREA     DS    CL(MQTMC_LENGTH)

```

Wizualna deklaracja podstawowa

```

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-nagłówek informacji o pracy

W poniższej tabeli podsumowano pola w strukturze.

Tabela 96. Pola w tabeli MQWIH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQWIH	StrucLength
<i>Encoding</i>	Kodowanie numeryczne danych, które są następujące: MQWIH	Kodowanie
<i>CodedCharSetId</i>	Identyfikator zestawu znaków dla danych, które są następujące: MQWIH	CodedCharSetId
<i>Format</i>	Nazwa formatu danych, które są następujące: MQWIH	Formatowanie

Tabela 96. Pola w tabeli MQWIH (kontynuacja)		
Pole	Opis	Temat
<i>Flags</i>	Flagi	<u>Flagi</u>
<i>ServiceName</i>	Nazwa usługi	<u>ServiceName</u>
<i>ServiceStep</i>	Nazwa kroku usługi	<u>ServiceStep</u>
<i>MsgToken</i>	Token komunikatu	<u>MsgToken</u>
<i>Reserved</i>	Zarezerwowane	<u>Zarezerwowane</u>

Przegląd produktu MQWIH

Struktura MQWIH opisuje informacje nagłówka dla komunikatu wysłanego do zarządzania obciążeniem programu z/OS (WLM). W przypadku dowolnej platformy IBM MQ obsługiwanej przez użytkownika można utworzyć i przelać komunikat zawierający strukturę MQWIH, ale tylko menedżer kolejek produktu IBM MQ for z/OS może współdziałać z menedżerem WLM. Oznacza to, że aby komunikat mógł zostać wysłany do menedżera kolejek z menedżerem kolejek innym niż z/OS, sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek produktu z/OS, przez który komunikat może być kierowany.

Dostępność: Wszystkie systemy IBM MQ oraz klienci IBM MQ połączone z tymi systemami.

Cel: Struktura MQWIH opisuje informacje, które muszą być obecne na początku komunikatu, który ma być obsługiwany przez menedżer obciążenia produktu z/OS.

Nazwa formatu: MQFMT_WORK_INFO_HEADER.

Zestaw znaków i kodowanie: pola w strukturze MQWIH znajdują się w zestawie znaków i kodowaniu podanym w polach *CodedCharSetId* i *Encoding* w strukturze nagłówka poprzedzającym MQWIH lub przez te pola w strukturze MQMD, jeśli wartość MQWIH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Użycie: Jeśli komunikat ma być przetworzony przez menedżera obciążenia produktu z/OS, komunikat musi rozpoczynać się od struktury MQWIH.

Pola dla MQWIH

Struktura MQWIH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CodedCharSetId (MQLONG)

Określa identyfikator zestawu znaków dla danych, które są zgodne ze strukturą MQWIH. Nie ma on zastosowania do danych znakowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT

Dane znakowe w danych po tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli wystąpi błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie można użyć tabeli MQCCSI_INHERIT, jeśli wartością pola *PutApplType* w deskrypcyjce MQMD jest MQAT_BROKER.

Początkowa wartość tego pola to MQCCSI_UNDEFINED.

Kodowanie (MQLONG)

Określa kodowanie numeryczne danych, które są zgodne ze strukturą MQWIH; nie ma zastosowania do danych liczbowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

Flags (MQLONG)

Wartość musi być następująca:

MQWIH_NONE

Brak flag.

Wartością początkową tego pola jest MQWIH_NONE.

Format (MQCHAR8)

Określa nazwę formatu danych, które są zgodne ze strukturą MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *Format* w strukturze MQMD.

Długość tego pola jest podana przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

MsgToken (MQBYTE16)

Jest to znacznik komunikatu, który jednoznacznie identyfikuje komunikat.

W przypadku wywołań MQPUT i MQPUT1 to pole jest ignorowane. Długość tego pola jest podana przez wartość MQ_MSG_TOKEN_LENGTH. Wartością początkową tego pola jest MQMTOK_NONE.

Zarezerwowane (MQCHAR32)

Jest to pole zastrzeżone. Musi być puste.

ServiceName (MQCHAR32)

Jest to nazwa usługi, która ma przetworzyć komunikat.

Długość tego pola jest podana przez wartość MQ_SERVICE_NAME_LENGTH. Początkowa wartość tego pola to 32 znaki puste.

ServiceStep (MQCHAR8)

Jest to nazwa kroku *ServiceName*, do którego odnosi się komunikat.

Długość tego pola jest podana przez wartość MQ_SERVICE_STEP_LENGTH. Początkowa wartość tego pola to 8 znaków odstępu.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQWIH_STRUC_ID

Identyfikator struktury nagłówka informacji o pracy.

Dla języka programowania C zdefiniowana jest również stała MQWIH_STRUC_ID_ARRAY; ma taką samą wartość jak MQWIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQWIH_STRUC_ID.

StrucLength (MQLONG)

Jest to długość struktury MQWIH. Wartość musi być następująca:

MQWIH_LENGTH_1

Długość struktury nagłówka informacji o pracy w wersji version-1.

Następująca stała określa długość bieżącej wersji:

MQWIH_CURRENT_LENGTH

Długość bieżącej wersji struktury nagłówka informacji o pracy.

Początkowa wartość tego pola to MQWIH_LENGTH_1.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQWIH_VERSION_1

Struktura nagłówka informacji o pracy Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQWIH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka informacji o pracy.

Początkowa wartość tego pola to MQWIH_VERSION_1.

Wartości początkowe i deklaracje języków dla produktu MQWIH

Tabela 97. Początkowe wartości pól w MQWIH dla MQWIH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQWIH_STRUC_ID	'WIH¬'
<i>Version</i>	MQWIH_VERSION_1	1
<i>StrucLength</i>	MQWIH_LENGTH_1	120
<i>Encoding</i>	Brak	0
<i>CodedCharSetId</i>	MQCCSI_UNDEFINED	0
<i>Format</i>	MQFMT_NONE	Puste
<i>Flags</i>	MQWIH_NONE	0
<i>ServiceName</i>	Brak	Puste
<i>ServiceStep</i>	Brak	Puste
<i>MsgToken</i>	MQMTOK_BRAK	Wartości null
<i>Reserved</i>	Brak	Puste
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		
2. W języku programowania C: zmienna makraParametr MQWIH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:		
<pre>MQWIH MyWIH = {MQWIH_DEFAULT};</pre>		

Deklaracja C

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQWIH structure */
    MQLONG     Encoding;         /* Numeric encoding of data that follows
                                MQWIH */
}
```

```

    MQLONG    CodedCharSetId; /* Character-set identifier of data that
                                follows MQWIH */
    MQCHAR8    Format;         /* Format name of data that follows
                                MQWIH */
    MQLONG    Flags;          /* Flags */
    MQCHAR32   ServiceName;    /* Service name */
    MQCHAR8    ServiceStep;    /* Service step name */
    MQBYTE16   MsgToken;       /* Message token */
    MQCHAR32   Reserved;       /* Reserved */
};

```

Deklaracja języka COBOL

```

**  MQWIH structure
10 MQWIH.
**    Structure identifier
15 MQWIH-STRUCID      PIC X(4).
**    Structure version number
15 MQWIH-VERSION     PIC S9(9) BINARY.
**    Length of MQWIH structure
15 MQWIH-STRUCLNGTH  PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQWIH
15 MQWIH-ENCODING    PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQWIH
15 MQWIH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQWIH
15 MQWIH-FORMAT      PIC X(8).
**    Flags
15 MQWIH-FLAGS       PIC S9(9) BINARY.
**    Service name
15 MQWIH-SERVICENAME  PIC X(32).
**    Service step name
15 MQWIH-SERVICESTEP  PIC X(8).
**    Message token
15 MQWIH-MSGTOKEN     PIC X(16).
**    Reserved
15 MQWIH-RESERVED     PIC X(32).

```

Deklaracja PL/I

```

dcl
1 MQWIH based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 StrucLength  fixed bin(31),    /* Length of MQWIH structure */
3 Encoding     fixed bin(31),    /* Numeric encoding of data that
                                follows MQWIH */
3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                that follows MQWIH */
3 Format        char(8),          /* Format name of data that follows
                                MQWIH */
3 Flags        fixed bin(31),    /* Flags */
3 ServiceName  char(32),         /* Service name */
3 ServiceStep  char(8),          /* Service step name */
3 MsgToken     char(16),         /* Message token */
3 Reserved     char(32);         /* Reserved */

```

Deklaracja High Level Assembler

MQWIH	DSECT	
MQWIH_STRUCID	DS	CL4 Structure identifier
MQWIH_VERSION	DS	F Structure version number
MQWIH_STRUCLNGTH	DS	F Length of MQWIH structure
MQWIH_ENCODING	DS	F Numeric encoding of data that follows MQWIH
* MQWIH_CODEDCHARSETID	DS	F Character-set identifier of data that follows MQWIH
* MQWIH_FORMAT	DS	CL8 Format name of data that follows MQWIH
MQWIH_FLAGS	DS	F Flags
MQWIH_SERVICENAME	DS	CL32 Service name
MQWIH_SERVICESTEP	DS	CL8 Service step name
MQWIH_MSGTOKEN	DS	XL16 Message token
MQWIH_RESERVED	DS	CL32 Reserved

```

*
MQWIH_LENGTH      EQU  *-MQWIH
                   ORG  MQWIH
MQWIH_AREA         DS   CL(MQWIH_LENGTH)

```

Wizualna deklaracja podstawowa

```

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-blok parametru procedury zewnętrznej

W poniższej tabeli podsumowano pola w strukturze.

Tabela 98. Pola w MQXP		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>ExitId</i>	Identyfikator wyjścia	ExitId
<i>ExitReason</i>	Przyczyna wywołania wyjścia	ExitReason
<i>ExitResponse</i>	Odpowiedź z wyjścia	ExitResponse
<i>ExitCommand</i>	Kod wywołania API	ExitCommand
<i>ExitParmCount</i>	Liczba parametrów	ExitParmLiczba
<i>ExitUserArea</i>	Obszar użytkownika	ObszarExitUser

Przegląd produktu MQXP

Dostępność: z/OS.

Cel: Struktura MQXP jest używana jako parametr wejścia/wyjścia do wyjścia funkcji API. Więcej informacji na temat tego wyjścia znajduje się w sekcji [Wyjście z funkcji API-przejście](#).

Zestaw znaków i kodowanie: Dane znakowe w produkcie MQXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Dane te są nadawane przez atrybut menedżera kolejek produktu *CodedCharSetId*. Dane liczbowe w MQXP są używane w rodzimym kodowaniu komputera. Dane te są podawane przez komendę MQENC_NATIVE.

Pola dla MQXP

Struktura MQXP zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ExitCommand (MQLONG)

To pole jest ustawiane przy wpisach do procedury wyjścia. Identyfikuje wywołanie API, które spowodowało wywołanie wyjścia:

MQXC_CALLBACK

Wywołanie CALLBACK.

MQXC_MQBACK

Wywołanie MQBACK.

MQXC_MQCB

Wywołanie MQCB.

MQXC_MQCLOSE

Wywołanie MQCLOSE.

MQXC_MQCMIT

Wywołanie MQCMIT.

MQXC_MQCTL

Wywołanie MQCTL.

MQXC_MQGET

Wywołanie MQGET.

MQXC_MQINQ

Wywołanie MQINQ.

MQXC_MQOPEN

Wywołanie MQOPEN.

MQXC_MQPUT

Wywołanie MQPUT.

MQXC_MQPUT1

Wywołanie MQPUT1 .

MQXC_MQSET

Wywołanie MQSET.

MQXC_MQSTAT

Wywołanie MQSTAT.

MQXC_MQSUB

Wywołanie MQSUB.

MQXC_MQSUBRQ

Wywołanie MQSUBRQ.

To jest pole wejściowe do wyjścia.

ExitId (MQLONG)

Wartość ta jest ustawiana przy wpisach do procedury wyjścia i wskazuje typ wyjścia:

MQXT_API_CROSSING_EXIT,

Wyjście funkcji API-wyjście dla CICS.

To jest pole wejściowe do wyjścia.

Liczba operacji ExitParm(MQLONG)

To pole jest ustawiane przy wpisach do procedury wyjścia. Zawiera ona liczbę parametrów, które są wymagane przez wywołanie programu MQ . Są to:

Nazwa połączenia	Liczba parametrów
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10

Nazwa połączenia	Liczba parametrów
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

To jest pole wejściowe do wyjścia.

ExitReason (MQLONG)

Ten parametr jest ustawiany podczas wprowadzania do procedury wyjścia. W przypadku wyjścia funkcji API, które wskazuje, czy procedura jest wywoływana przed lub po wykonaniu wywołania API, należy:

MQXR_PRZED

Przed wykonaniem interfejsu API.

MQXR_AFTER

Po wykonaniu interfejsu API.

To jest pole wejściowe do wyjścia.

ExitResponse (MQLONG)

Wartość ta jest ustawiana przez wyjście w celu komunikowania się z programem wywołującym. Zdefiniowane są następujące wartości:

MQXCC_OK

Wyjście zostało zakończone pomyślnie.

MQXCC_SUPPRESS_FUNCTION

Funkcja pomijania.

Jeśli ta wartość jest ustawiona przez wyjście funkcji API o nazwie *przed* wywołaniem API, wywołanie API nie jest wykonywane. *CompCode* dla wywołania jest ustawiony na MQCC_FAILED, *Reason* jest ustawiony na MQRC_SUPPRESSED_BY_EXIT, a pozostałe parametry pozostają tak samo, jak wyjście pozostawiane przez wyjście.

Jeśli ta wartość jest ustawiona przez wyjście funkcji API o nazwie *po* wywołaniu interfejsu API, jest ono ignorowane przez menedżer kolejek.

MQXCC_SKIP_FUNCTION,

Funkcja pomijania.

Jeśli ta wartość jest ustawiona przez wyjście funkcji API o nazwie *przed* wywołaniem API, wywołanie API nie jest wykonywane; *CompCode* i *Reason* oraz wszystkie pozostałe parametry pozostają jako wyjście pozostawione przez interfejs API.

Jeśli ta wartość jest ustawiona przez wyjście funkcji API o nazwie *po* wywołaniu interfejsu API, jest ono ignorowane przez menedżer kolejek.

To jest pole wyjściowe z wyjścia.

Obszar ExitUser(MQBYTE16)

Jest to pole, które jest dostępne dla wyjścia do użycia. Jest on inicjowany do zera binarnego dla długości pola przed pierwszym wywołaniem wyjścia dla zadania, a następnie wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane w wywołaniach wyjścia. Zdefiniowana jest następująca wartość:

MQXUA_NONE

Brak informacji o użytkowniku.

Wartość jest binarna zero dla długości pola.

Dla języka programowania C zdefiniowana jest również stała MQXUA_NONE_ARRAY; ma ona taką samą wartość jak MQXUA_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_EXIT_USER_AREA_LENGTH. Jest to pole wejściowe/ wyjściowe do wyjścia.

Zarezerwowane (MQLONG)

Jest to pole zastrzeżone. Jego wartość nie jest istotna dla wyjścia.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

Identyfikator MQXP_STRUC_ID

Identyfikator struktury parametru wyjścia.

Dla języka programowania C jest również zdefiniowana stała zmienna MQXP_STRUC_ID_ARRAY; ta sama wartość ma wartość MQXP_STRUC_ID, ale jest to tablica znaków zamiast łańcucha.

To jest pole wejściowe do wyjścia.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQXP_VERSION_1

Numer wersji dla parametru wyjścia-struktura bloku.

Uwaga: Gdy zostanie wprowadzona nowa wersja tej struktury, układ istniejącej części nie jest zmieniany. W związku z tym wyjście musi sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji, która zawiera pola, które mają być używane przez program obsługi wyjścia.

To jest pole wejściowe do wyjścia.

Deklaracje językowe

Ta struktura jest obsługiwana w następujących językach programowania.

Deklaracja C

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     ExitId;            /* Exit identifier */
    MQLONG     ExitReason;        /* Reason for invocation of exit */
    MQLONG     ExitResponse;      /* Response from exit */
    MQLONG     ExitCommand;       /* API call code */
    MQLONG     ExitParmCount;     /* Parameter count */
    MQLONG     Reserved;         /* Reserved */
    MQBYTE16   ExitUserArea;     /* User area */
};
```

Deklaracja języka COBOL

```
**      MQXP structure
10 MQXP.
**      Structure identifier
15 MQXP-STRUCID      PIC X(4).
**      Structure version number
15 MQXP-VERSION      PIC S9(9) BINARY.
**      Exit identifier
15 MQXP-EXITID       PIC S9(9) BINARY.
**      Reason for invocation of exit
15 MQXP-EXITREASON   PIC S9(9) BINARY.
**      Response from exit
15 MQXP-EXITRESPONSE PIC S9(9) BINARY.
**      API call code
```

```

15 MQXP-EXITCOMMAND PIC S9(9) BINARY.
**   Parameter count
15 MQXP-EXITPARMCOUNT PIC S9(9) BINARY.
**   Reserved
15 MQXP-RESERVED PIC S9(9) BINARY.
**   User area
15 MQXP-EXITUSERAREA PIC X(16).

```

Deklaracja PL/I

```

dcl
1 MQXP based,
3 StrucId      char(4),          /* Structure identifier */
3 Version      fixed bin(31),    /* Structure version number */
3 ExitId       fixed bin(31),    /* Exit identifier */
3 ExitReason   fixed bin(31),    /* Reason for invocation of exit */
3 ExitResponse fixed bin(31),    /* Response from exit */
3 ExitCommand  fixed bin(31),    /* API call code */
3 ExitParmCount fixed bin(31),   /* Parameter count */
3 Reserved     fixed bin(31),    /* Reserved */
3 ExitUserArea char(16);        /* User area */

```

Deklaracja High Level Assembler

```

MQXP          DSECT
MQXP_STRUCID  DS CL4   Structure identifier
MQXP_VERSION  DS F     Structure version number
MQXP_EXITID   DS F     Exit identifier
MQXP_EXITREASON DS F   Reason for invocation of exit
MQXP_EXITRESPONSE DS F Response from exit
MQXP_EXITCOMMAND DS F  API call code
MQXP_EXITPARMCOUNT DS F Parameter count
MQXP_RESERVED DS F    Reserved
MQXP_EXITUSERAREA DS XL16 User area
*
MQXP_LENGTH   EQU *-MQXP
ORG MQXP
MQXP_AREA     DS CL(MQXP_LENGTH)

```

MQXQH-nagłówek kolejki transmisji

W poniższej tabeli podsumowano pola w strukturze.

Tabela 99. Pola w MQXQH		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>RemoteQName</i>	Nazwa kolejki docelowej	RemoteQName
<i>RemoteQMgrName</i>	Nazwa docelowego menedżera kolejek	RemoteQMgrNazwa
<i>MsgDesc</i>	Oryginalny deskryptor komunikatu	MsgDesc

Przegląd produktu MQXQH

Dostępność: wszystkie systemy IBM MQ i klienci IBM MQ .

Cel: Struktura MQXQH opisuje informacje, które są poprzedzane danymi komunikatów aplikacji komunikatów, gdy znajdują się one w kolejkach transmisji. Kolejka transmisji jest specjalnym typem kolejki lokalnej, która tymczasowo przechowuje komunikaty przeznaczone dla kolejek zdalnych (czyli jest przeznaczone dla kolejek, które nie należą do lokalnego menedżera kolejek). Kolejka transmisji jest oznaczana za pomocą atrybutu kolejki *Usage* o wartości MQUS_TRANSMISSION.

Nazwa formatu: MQFMT_XMIT_Q_HEADER.

Zestaw znaków i kodowanie: Dane w tabeli MQXQH muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez komendę MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie dla tabeli MQXQH w polach *CodedCharSetId* i *Encoding* w:

- Osobny deskryptor MQMD (jeśli struktura MQXQH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQXQH (wszystkie inne obserwacje).

Użycie: Komunikat, który znajduje się w kolejce transmisji, ma dwa deskryptory komunikatów:

- Jeden deskryptor komunikatu jest przechowywany oddzielnie od danych komunikatu, jest nazywany *odrębnym deskryptorem komunikatu* jest generowany przez menedżer kolejek, gdy komunikat jest umieszczany w kolejce transmisji. Niektóre pola w oddzielnym deskrypcorze komunikatu są kopiowane z deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Osobny deskryptor komunikatu jest to ten, który jest zwracany do aplikacji w parametrze *MsgDesc* wywołania MQGET, gdy komunikat jest usuwany z kolejki transmisji.

- Drugi deskryptor komunikatu jest przechowywany w strukturze MQXQH jako część danych komunikatu. Jest to nazywane *osadzonym deskryptorem komunikatu* jest kopią deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 (z niewielkimi zmianami).

Osadzony deskryptor komunikatu jest zawsze MQMD w wersji version-1 . Jeśli komunikat umieszczony przez aplikację ma wartości inne niż domyślne dla co najmniej jednej z pól version-2 w strukturze MQMD, struktura MQMDE jest zgodna z tabelą MQXQH, po czym następuje po kolei dane komunikatu aplikacji (jeśli istnieją). MQMDE:

- Wygenerowane przez menedżer kolejek (jeśli aplikacja używa deskryptora MQMD version-2 w celu umieszczenia komunikatu), lub
- Jest już obecny na początku danych komunikatu aplikacji (jeśli aplikacja używa deskryptora MQMD w wersji version-1 do umieszczenia komunikatu).

Osadzony deskryptor komunikatu jest tym, który jest zwracany do aplikacji w parametrze *MsgDesc* wywołania MQGET, gdy komunikat jest usuwany z końcowej kolejki docelowej.

Pola w osobnym deskrypcorze komunikatu: pola w oddzielnym deskrypcorze komunikatu są ustawiane przez menedżer kolejek zgodnie z wyświetleniem. Jeśli menedżer kolejek nie obsługuje deskryptora MQMD z wersji version-2 , nie jest używana funkcja MQMD w wersji version-1 bez utraty funkcji.

Pole w oddzielnej tabeli MQMD	Użyta wartość
<i>StrucId</i>	MQMD_STRUC_ID
<i>Version</i>	MQMD_VERSION_2
<i>Report</i>	Skopiowano z osadzonego deskryptora komunikatu, ale z bitami określonymi przez parametr MQRO_ACCEPT_UNSUP_IF_XMIT_MASK ustawiono na zero. (uniemożliwia to wygenerowanie komunikatu raportu COA lub COD, gdy komunikat jest umieszczany w kolejce transmisji lub usuwany z kolejki transmisji).
<i>MsgType</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>Expiry</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>Feedback</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>Encoding</i>	MQENC_NATIVE (patrz uwaga)
<i>CodedCharSetId</i>	Atrybut <i>CodedCharSetId</i> menedżera kolejek.
<i>Format</i>	MQFMT_XMIT_Q_HEADER
<i>Priority</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>Persistence</i>	Skopiowano z osadzonego deskryptora komunikatu.

Pole w oddzielnej tabeli MQMD	Użyta wartość
<i>MsgId</i>	Nowa wartość jest generowana przez menedżer kolejek. Ten identyfikator komunikatu różni się od <i>MsgId</i> , który menedżer kolejek mógł wygenerować dla opisanego wcześniej osadzonego deskryptora komunikatu.
<i>CorrelId</i>	<i>MsgId</i> z osadzonego deskryptora komunikatu. W przypadku komunikatów umieszczanych w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE, <i>CorrelId</i> jest zarezerwowana do użytku wewnętrznego.
<i>BackoutCount</i>	0
<i>ReplyToQ</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>ReplyToQMgr</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>UserIdentifier</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>AccountingToken</i>	Skopiowano z osadzonego deskryptora komunikatu. W przypadku komunikatów umieszczanych w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE, <i>AccountingToken</i> jest zarezerwowana do użytku wewnętrznego.
<i>ApplIdentityData</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>PutApplType</i>	MQAT_QMGR
<i>PutApplName</i>	Pierwsze 28 bajtów nazwy menedżera kolejek.
<i>PutDate</i>	Data umieszczenia komunikatu w kolejce transmisji.
<i>PutTime</i>	Czas umieszczenia komunikatu w kolejce transmisji.
<i>ApplOriginData</i>	Puste
<i>GroupId</i>	MQGI_NONE
<i>MsgSeqNumber</i>	1
<i>Offset</i>	0
<i>MsgFlags</i>	MQMF_NONE
<i>OriginalLength</i>	MQOL_NIEZDEFINIOWANY

- W systemie Windows wartość MQENC_NATIVE dla programu Micro Focus COBOL różni się od wartości dla C. Wartość w polu *Encoding* w osobnym deskrypcorze komunikatu jest zawsze wartością dla języka C w tych środowiskach. Wartość ta wynosi 546 w postaci dziesiętnej. Ponadto pola liczb całkowitych w strukturze MQXQH znajdują się w kodowaniu, które odpowiada tej wartości (rodzime kodowanie Intel).

Pola w osadzonym deskrypcorze komunikatu: pola w osadzonym deskrypcorze komunikatu mają te same wartości, co wartości w parametrze *MsgDesc* wywołania MQPUT lub MQPUT1 , z wyjątkiem następujących:

- W polu *Version* zawsze znajduje się wartość MQMD_VERSION_1.
- Jeśli pole *Priority* ma wartość MQPRI_PRIORITY_AS_Q_DEF, to jest ono zastępowane wartością atrybutu *DefPriority* kolejki.
- Jeśli pole *Persistence* ma wartość MQPER_PERSISTENCE_AS_Q_DEF, to jest ona zastępowana wartością atrybutu *DefPersistence* kolejki.
- Jeśli pole *MsgId* ma wartość MQMI_NONE lub podano opcję MQPMO_NEW_MSG_ID lub komunikat jest komunikatem listy dystrybucyjnej, *MsgId* jest zastępowany przez nowy identyfikator komunikatu wygenerowany przez menedżer kolejek.

Gdy komunikat z listą dystrybucyjną jest dzielony na mniejsze komunikaty listy dystrybucyjnej umieszczone w różnych kolejkach transmisji, pole *MsgId* w każdym nowym osadzonym deskrytorze komunikatów jest takie samo, jak w oryginalnym komunikacie listy dystrybucyjnej.

- Jeśli została określona opcja MQPMO_NEW_CORREL_ID, *CorrelId* jest zastępowana nowym identyfikatorem korelacji wygenerowanym przez menedżer kolejek.
- Pola kontekstu są ustawiane zgodnie z opcjami MQPMO_*_CONTEXT określonymi w parametrze *PutMsgOpts*. Pola kontekstu są następujące:
 - *AccountingToken*
 - *ApplIdentityData*
 - *ApplOriginData*
 - *PutApplName*
 - *PutApplType*
 - *PutDate*
 - *PutTime*
 - *UserIdentifier*
- Pola version-2 (jeśli były obecne) są usuwane z deskryptora MQMD, a następnie przenoszone do struktury MQMDE, jeśli co najmniej jedna z pól version-2 ma wartość niedomyślną.

umieszczanie komunikatów w kolejkach zdalnych: gdy aplikacja umieszcza komunikat w kolejce zdalnej (poprzez podanie nazwy kolejki zdalnej bezpośrednio lub przy użyciu lokalnej definicji kolejki zdalnej), lokalny menedżer kolejek:

- Tworzy strukturę MQXQH zawierającą osadzony deskryptor komunikatu
- Dodaje MQMDE, jeśli jest potrzebny i nie jest jeszcze obecny.
- Dołącza dane komunikatu aplikacji
- Umieszcza komunikat w odpowiedniej kolejce transmisji

Umieszczanie komunikatów bezpośrednio w kolejkach transmisji: aplikacja może również umieścić komunikat bezpośrednio w kolejce transmisji. W takim przypadku aplikacja musi poprzedzić dane komunikatu aplikacji ze strukturą MQXQH i zainicjalizować pola odpowiednimi wartościami. Oprócz tego pole *Format* w parametrze *MsgDesc* wywołania MQPUT lub MQPUT1 musi mieć wartość MQFMT_XMIT_Q_HEADER.

Dane znakowe w strukturze MQXQH utworzonej przez aplikację muszą znajdować się w zestawie znaków lokalnego menedżera kolejek (zdefiniowanym przez atrybut *CodedCharSetId* menedżera kolejek), a dane całkowite muszą znajdować się w rodzimym kodowaniu komputera. Ponadto dane znakowe w strukturze MQXQH muszą być dopełniane spacjami do zdefiniowanej długości pola; dane nie mogą być kończone przedwcześnie za pomocą znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca wartości NULL i kolejnych znaków w puste miejsca w strukturze MQXQH.

Menedżer kolejek nie sprawdza jednak, czy struktura MQXQH jest obecna lub czy określono poprawne wartości dla pól.

Aplikacje nie powinny umieszczać swoich komunikatów bezpośrednio w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Pobieranie komunikatów z kolejek transmisji: aplikacje, które pobierają komunikaty z kolejki transmisji, muszą przetwarzać informacje w strukturze MQXQH w odpowiedni sposób. Obecność struktury MQXQH na początku danych komunikatu aplikacji jest wskazywana przez wartość MQFMT_XMIT_Q_HEADER, która jest zwracana w polu *Format* w parametrze *MsgDesc* wywołania MQGET. Wartości zwracane w polach *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* wskazują zestaw znaków i kodowanie danych znakowych i całkowitoliczbowych w strukturze MQXQH. Zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane za pomocą pól *CodedCharSetId* i *Encoding* w osadzonym deskrytorze komunikatu.

Pola dla MQXQH

Struktura MQXQH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

MsgDesc (MQMD1)

Jest to osadzony deskryptor komunikatu i jest to bliska kopia deskryptora komunikatu MQMD, która została określona jako parametr *MsgDesc* w wywołaniu MQPUT lub MQPUT1, gdy komunikat został pierwotnie umieszczony w kolejce zdalnej.

Uwaga: Jest to deskryptor MQMD w wersji version-1.

Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQMD.

Nazwa RemoteQMGr(MQCHAR48)

Jest to nazwa menedżera kolejek lub grupy współużytkowania kolejki, która jest właścicielem kolejki, która jest jawnym miejscem docelowym dla komunikatu.

Jeśli komunikat jest komunikatem z listą dystrybucyjną, pole *RemoteQMGrName* jest puste.

Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

RemoteQName (MQCHAR48)

Jest to nazwa kolejki komunikatów, która jest pozornym miejscem docelowym dla komunikatu (może to okazać się, że nie jest to docelowe miejsce docelowe, jeśli na przykład kolejka ta jest zdefiniowana w produkcie *RemoteQMGrName* jako lokalna definicja innej kolejki zdalnej).

Jeśli komunikat jest komunikatem listy dystrybucyjnej (to znaczy pole *Format* w deskrypcorze osadzonego komunikatu to MQFMT_DIST_HEADER), pole *RemoteQName* jest puste.

Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania.

StrucId (MQCHAR4)

Jest to identyfikator struktury. Wartość musi być następująca:

MQXQH_STRUC_ID

Identyfikator struktury nagłówka kolejki transmisji.

Dla języka programowania C jest również zdefiniowana stała MQXQH_STRUC_ID_ARRAY; ta sama wartość ma taką samą wartość jak MQXQH_STRUC_ID, ale jest to tablica znaków zamiast łańcucha.

Wartością początkową tego pola jest MQXQH_STRUC_ID.

Wersja (MQLONG)

Jest to numer wersji struktury. Wartość musi być następująca:

MQXQH_VERSION_1

Numer wersji struktury nagłówka kolejki transmisji.

Następująca stała określa numer wersji bieżącej wersji:

MQXQH_CURRENT_VERSION

Bieżąca wersja struktury nagłówka kolejki transmisji.

Początkowa wartość tego pola to MQXQH_VERSION_1.

Wartości początkowe i deklaracje języków dla MQXQH

Tabela 100. Początkowe wartości pól w MQXQH dla MQXQH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQXQH_STRUC_ID	'XQH'

Tabela 100. Początkowe wartości pól w MQXQH dla MQXQH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
Version	MQXQH_VERSION_1	1
RemoteQName	Brak	Pusty łańcuch lub odstępy
RemoteQMgrName	Brak	Pusty łańcuch lub odstępy
MsgDesc	Te same nazwy i te same wartości co MQMD; patrz Tabela 55 na stronie 467	-

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makra MQXQH_DEFAULT zawiera wartości, które są wymienione w tabeli. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48    RemoteQName;     /* Name of destination queue */
    MQCHAR48    RemoteQMgrName; /* Name of destination queue manager */
    MQMD1      MsgDesc;          /* Original message descriptor */
};
```

Deklaracja języka COBOL

```
** MQXQH structure
10 MQXQH.
** Structure identifier
15 MQXQH-STRUCID PIC X(4).
** Structure version number
15 MQXQH-VERSION PIC S9(9) BINARY.
** Name of destination queue
15 MQXQH-REMOTEQNAME PIC X(48).
** Name of destination queue manager
15 MQXQH-REMOTEQMGRNAME PIC X(48).
** Original message descriptor
15 MQXQH-MSGDESC.
** Structure identifier
20 MQXQH-MSGDESC-STRUCID PIC X(4).
** Structure version number
20 MQXQH-MSGDESC-VERSION PIC S9(9) BINARY.
** Report options
20 MQXQH-MSGDESC-REPORT PIC S9(9) BINARY.
** Message type
20 MQXQH-MSGDESC-MSGTYPE PIC S9(9) BINARY.
** Expiry time
20 MQXQH-MSGDESC-EXPIRY PIC S9(9) BINARY.
** Feedback or reason code
20 MQXQH-MSGDESC-FEEDBACK PIC S9(9) BINARY.
** Numeric encoding of message data
20 MQXQH-MSGDESC-ENCODING PIC S9(9) BINARY.
** Character set identifier of message data
20 MQXQH-MSGDESC-CODEDCHARSETID PIC S9(9) BINARY.
** Format name of message data
20 MQXQH-MSGDESC-FORMAT PIC X(8).
** Message priority
20 MQXQH-MSGDESC-PRIORITY PIC S9(9) BINARY.
```



```

**      Message persistence
20 MQXQH-MSGDESC-PERSISTENCE      PIC S9(9) BINARY.
**      Message identifier
20 MQXQH-MSGDESC-MSGID            PIC X(24).
**      Correlation identifier
20 MQXQH-MSGDESC-CORRELID        PIC X(24).
**      Backout counter
20 MQXQH-MSGDESC-BACKOUTCOUNT    PIC S9(9) BINARY.
**      Name of reply-to queue
20 MQXQH-MSGDESC-REPLYTOQ        PIC X(48).
**      Name of reply queue manager
20 MQXQH-MSGDESC-REPLYTOQMGR      PIC X(48).
**      User identifier
20 MQXQH-MSGDESC-USERIDENTIFIER   PIC X(12).
**      Accounting token
20 MQXQH-MSGDESC-ACCOUNTINGTOKEN  PIC X(32).
**      Application data relating to identity
20 MQXQH-MSGDESC-APPLIDENTITYDATA PIC X(32).
**      Type of application that put the message
20 MQXQH-MSGDESC-PUTAPPLTYPE      PIC S9(9) BINARY.
**      Name of application that put the message
20 MQXQH-MSGDESC-PUTAPPLNAME      PIC X(28).
**      Date when message was put
20 MQXQH-MSGDESC-PUTDATE          PIC X(8).
**      Time when message was put
20 MQXQH-MSGDESC-PUTTIME          PIC X(8).
**      Application data relating to origin
20 MQXQH-MSGDESC-APPLORIGINDATA   PIC X(4).

```

Deklaracja PL/I

```

dcl
  1 MQXQH based,
    3 StrucId          char(4),          /* Structure identifier */
    3 Version          fixed bin(31),    /* Structure version number */
    3 RemoteQName      char(48),         /* Name of destination queue */
    3 RemoteQMgrName   char(48),         /* Name of destination queue
                                          manager */
    3 MsgDesc,
      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 UserId         char(12),         /* User identifier */
      5 AccountingToken char(32),        /* Accounting token */
      5 ApplIdentityData char(32),       /* Application data relating to
                                          identity */
      5 PutApplType    fixed bin(31),    /* Type of application that put the
                                          message */
      5 PutApplName    char(28),         /* Name of application that put the
                                          message */
      5 PutDate        char(8),          /* Date when message was put */
      5 PutTime        char(8),          /* Time when message was put */
      5 ApplOriginData char(4);          /* Application data relating to
                                          origin */

```

Deklaracja High Level Assembler

MQXQH	DSECT	
MQXQH_STRUCID	DS	CL4 Structure identifier
MQXQH_VERSION	DS	F Structure version number
MQXQH_REMOTEQNAME	DS	CL48 Name of destination queue

MQXQH_REMOTEQMGRNAME	DS	CL48	Name of destination queue manager
*MQXQH_MSGDESC	DS	0F	Force fullword alignment
MQXQH_MSGDESC_STRUCID	DS	CL4	Structure identifier
MQXQH_MSGDESC_VERSION	DS	F	Structure version number
MQXQH_MSGDESC_REPORT	DS	F	Report options
MQXQH_MSGDESC_MSGTYPE	DS	F	Message type
MQXQH_MSGDESC_EXPIRY	DS	F	Expiry time
MQXQH_MSGDESC_FEEDBACK	DS	F	Feedback or reason code
MQXQH_MSGDESC_ENCODING	DS	F	Numeric encoding of message data
*MQXQH_MSGDESC_CODEDCHARSETID	DS	F	Character set identifier of message data
*MQXQH_MSGDESC_FORMAT	DS	CL8	Format name of message data
MQXQH_MSGDESC_PRIORITY	DS	F	Message priority
MQXQH_MSGDESC_PERSISTENCE	DS	F	Message persistence
MQXQH_MSGDESC_MSGID	DS	XL24	Message identifier
MQXQH_MSGDESC_CORRELID	DS	XL24	Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT	DS	F	Backout counter
MQXQH_MSGDESC_REPLYTOQ	DS	CL48	Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR	DS	CL48	Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER	DS	CL12	User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN	DS	XL32	Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA	DS	CL32	Application data relating to identity
*MQXQH_MSGDESC_PUTAPPLTYPE	DS	F	Type of application that put the message
*MQXQH_MSGDESC_PUTAPPLNAME	DS	CL28	Name of application that put the message
*MQXQH_MSGDESC_PUTDATE	DS	CL8	Date when message was put
MQXQH_MSGDESC_PUTTIME	DS	CL8	Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA	DS	CL4	Application data relating to origin
*MQXQH_MSGDESC_LENGTH	EQU	*-MQXQH_MSGDESC	
	ORG	MQXQH_MSGDESC	
MQXQH_MSGDESC_AREA	DS	CL(MQXQH_MSGDESC_LENGTH)	
*MQXQH_LENGTH	EQU	*-MQXQH	
	ORG	MQXQH	
MQXQH_AREA	DS	CL(MQXQH_LENGTH)	

Wizualna deklaracja podstawowa

```

Type MQXQH
  StrucId      As String*4  'Structure identifier'
  Version      As Long      'Structure version number'
  RemoteQName  As String*48 'Name of destination queue'
  RemoteQMGRName As String*48 'Name of destination queue manager'
  MsgDesc      As MQMD1     'Original message descriptor'
End Type

```

Wywołania funkcji

Ta sekcja zawiera informacje na temat wszystkich wywołań MQI, które są możliwe. Opisy, składnia, informacje o parametrach, uwagi dotyczące użycia i wywołania językowe dla każdego możliwego języka są podane dla każdego z różnych wywołań.

Opisy wywołań

W tej sekcji opisano wywołania MQI.

- [“MQBACK-wycofanie zmian” na stronie 633](#)
- [“MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 637](#)
- [“MQBUFMH-przekształcanie buforu w uchwyt komunikatu” na stronie 640](#)
- [“MQCB-zarządzanie wywołaniem zwrotnym” na stronie 644](#)
- [“MQCB_FUNCTION-funkcja Callback” na stronie 654](#)
- [“MQCLOSE-zamknięcie obiektu” na stronie 655](#)
- [“MQCMIT-zatwierdzanie zmian” na stronie 664](#)

- [“MQCONN-Połączenie menedżera kolejek” na stronie 668](#)
- [“MQCONN-Connect menedżer kolejek \(rozszerzony\)” na stronie 676](#)
- [“MQCRTMH-Tworzenie uchwytu komunikatu” na stronie 682](#)
- [“MQCTL-wywołania zwrotne sterowania” na stronie 686](#)
- [“MQDISC-rozłączenie menedżera kolejek” na stronie 692](#)
- [“MQDLTMH-usuwanie uchwytu komunikatu” na stronie 696](#)
- [“MQDLTMP-usunięcie właściwości komunikatu” na stronie 699](#)
- [“MQGET-Pobieranie komunikatu” na stronie 701](#)
- [“MQINQ-zapytanie o atrybuty obiektu” na stronie 714](#)
- [“MQINQMMP-właściwość komunikatu Inquire” na stronie 732](#)
- [“MQMHBUF-Przekształć uchwyt komunikatu w bufor” na stronie 738](#)
- [“MQOPEN-obiekt otwarty” na stronie 742](#)
- [“MQPUT-umieszczanie komunikatu” na stronie 760](#)
- [“MQPUT1 -Umieść jeden komunikat” na stronie 774](#)
- [“MQSET-ustawienie atrybutów obiektu” na stronie 785](#)
- [“MQSETMP-ustawienie właściwości komunikatu” na stronie 791](#)
- [“MQSTAT-pobieranie informacji o statusie” na stronie 796](#)
- [“MQMHBUF-Przekształć uchwyt komunikatu w bufor” na stronie 738](#)
- [“MQSUB-Zarejestruj subskrypcję” na stronie 800](#)
- [“MQSUBRQ-żądanie subskrypcji” na stronie 807](#)

Pomoc elektroniczna na temat Platformy UNIX, w postaci *man* stron, jest dostępna dla tych połączeń.

Uwaga: Wywołania powiązane z konwersją danych, MQXCNCV i MQ_DATA_CONV_EXIT znajdują się w produkcie [“Konwersja danych” na stronie 919](#).

Konwencje używane w opisach wywołań

Dla każdego wywołania ta kolekcja tematów zawiera opis parametrów i użycia wywołania w formacie, który jest niezależny od języka programowania. Następuje to po typowych wywołaniach wywołania oraz typowych deklaracjach jego parametrów w każdym z obsługiwanych języków programowania.

Ważne: Podczas kodowania wywołań interfejsu API produktu IBM MQ należy upewnić się, że zostały podane wszystkie odpowiednie parametry (opisane w poniższych sekcjach). Niewykonalne działanie może spowodować nieprzewidywalne rezultaty.

Opis każdego wywołania zawiera następujące sekcje:

Nazwa połączenia

Nazwa połączenia, po której następuje krótki opis celu wywołania.

Parametry

W przypadku każdego parametru po nazwie występuje jego typ danych w nawiasach () oraz jedną z następujących czynności:

wejściowe,

Informacje są wyświetlane w parametrze, gdy użytkownik wywoła połączenie.

wyniki

Menedżer kolejek zwraca informacje w parametrze, gdy wywołanie zakończy się lub nie powiedzie się.

Wejście/wyjście

Informacje są wprowadzane w parametrze podczas wykonywania wywołania, a menedżer kolejek zmienia informacje, gdy wywołanie zakończy się lub zakończy się niepowodzeniem.

Na przykład:

Compcode (MQLONG)-dane wyjściowe

W niektórych przypadkach typ danych to struktura. We wszystkich przypadkach istnieje więcej informacji na temat typu danych lub struktury w produkcie [“Elementarne typy danych” na stronie 237](#).

Ostatnie dwa parametry w każdym wywołaniu to kod zakończenia i kod przyczyny. Kod zakończenia wskazuje, czy wywołanie zostało zakończone pomyślnie, częściowo, czy nie. Dodatkowe informacje na temat częściowego powodzenia lub niepowodzenia wywołania są podane w kodzie przyczyny. Więcej informacji na temat każdego kodu zakończenia i przyczyny można znaleźć w sekcji [“Kody powrotu” na stronie 885](#).

Użycie notatek

Dodatkowe informacje na temat połączenia, opisujące, jak go używać oraz wszelkie ograniczenia w jego stosowaniu.

Wywołanie języka asemblera

Typowe wywołanie wywołania oraz deklaracja jego parametrów w języku asemblera.

Wywołanie C

Typowe wywołanie wywołania, oraz deklaracja jej parametrów, w C.

Wywołanie języka COBOL

Typowe wywołanie wywołania i deklaracja jego parametrów w języku COBOL.

Wywołanie PL/I

Typowe wywołanie wezwania oraz deklaracja jej parametrów, w PL/I.

Wszystkie parametry są przekazywane przez referencję.

Wywołanie języka Visual Basic

Typowe wywołanie wywołania oraz deklaracja jego parametrów w Visual Basic.

Inne konwencje notacji to:

State

Nazwy stałych są wyświetlane wielkimi literami, na przykład: MQOO_OUTPUT. Zestaw stałych o tym samym przedrostku jest przedstawiony w następujący sposób: MQIA_*. Wartość stałej znajduje się w sekcji [“State” na stronie 61](#).

Tablice

W niektórych wywołaniach parametry są tablicami łańcuchów znaków, które nie mają stałych rozmiarów. W opisach tych parametrów małe n reprezentuje stałą numeryczną. Po zakodowaniu deklaracji dla tego parametru należy zastąpić n wartością numeryczną, która jest wymagana.

Korzystanie z połączeń w języku C

Parametry, które są *tylko danymi wejściowymi* i typu MQHCONN, MQHOBJ, MQHMSG lub MQLONG, są przekazywane przez wartość. W przypadku wszystkich pozostałych parametrów parametr *adres* parametru jest przekazywany przez wartość.

Nie ma potrzeby określania wszystkich parametrów, które są przekazywane przez adres za każdym razem, gdy wywoływana jest funkcja. Jeśli dany parametr nie jest wymagany, należy określić pusty wskaźnik jako parametr w wywołaniu funkcji, w miejsce adresu danych parametrów. Parametry, dla których jest to możliwe, są identyfikowane w opisach wywołań.

Jako wartość wywołania (w terminologii C) nie jest zwracany żaden parametr, oznacza to, że wszystkie wywołania zwracają wartość void.

Deklarowanie parametru buforu

W przypadku wywołań MQGET, MQPUT i MQPUT1 każdy z nich ma jeden parametr, który ma niezdefiniowany typ danych: parametr *Buffer*. Ten parametr służy do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są przedstawione w przykładach C jako tablice MQBYTE. Parametry można deklarować w ten sposób, ale zwykle wygodniejsze jest zadeklarowanie ich jako konkretnej struktury,

która opisuje układ danych w komunikacie. Prototyp funkcji deklaruje parametr jako wskaźnik-do-void, tak aby można było określić adres dowolnego rodzaju danych jako parametru w wywołaniu wywołania.

Wskaźnik-do-void jest wskaźnikiem do danych o niezdefiniowanym formacie. Jest on zdefiniowany jako:

```
typedef void *PMQVOID;
```

MQBACK-wycofanie zmian

Wywołanie MQBACK wskazuje menedżerowi kolejek, że wszystkie operacje pobierania i umieszczania komunikatów wystąpiły od ostatniego punktu synchronizacji, dla którego ma zostać wykonana kopia zapasowa.

Komunikaty umieszczone jako część jednostki pracy są usuwane; komunikaty pobrane jako część jednostki pracy są ponownie umieszczane w kolejce.

- W systemie z/OS wywołanie to jest używane tylko przez programy wsadowe (w tym IMS wsadowe programy DL/I).
- W systemie IBM i wywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Składnia

MQBACK (*Hconn*, *Compcode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

Kod obliczeniowy

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_OUTCOME_PENDING

(2124, X'84C') Wynik operacji back-out jest w toku.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

Błąd MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwył połączenia nie jest poprawny.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

MQRC_OUTCOME_MIXED

(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Tego wywołania można użyć tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Może to być:

- Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby MQ.
- Globalna jednostka pracy, w której zmiany mogą wpływać na zasoby należące do innych menedżerów zasobów, a także wpływają na zasoby MQ.

Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja [“MQBEGIN-Rozpoczęcie jednostki pracy”](#) na stronie 637.

2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiednich wywołań zwrotnych zamiast MQBACK. Środowisko może również obsługiwać niejawne wycofania spowodowane przez nieprawidłowe zakończenie działania aplikacji.
- W systemie z/OS należy użyć następujących wywołań:
 - Programy wsadowe (w tym wsadowe programy DL/I produktu IMS) mogą używać wywołania MQBACK, jeśli jednostka pracy ma wpływ tylko na zasoby produktu MQ. Jeśli jednak jednostka

pracy ma wpływ na zasoby zarówno MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnionego przez usługę RRS (Recoverable Resource Service) produktu z/OS . Wywołanie SRRBACK powoduje wycofania zmian w zasobach należących do menedżerów zasobów, które zostały włączone dla koordynacji RRS.

- Aplikacje produktu CICS muszą używać komendy EXEC CICS SYNCPOINT ROLLBACK do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
 - Aplikacje produktu IMS (inne niż wsadowe programy DL/I) muszą używać wywołań programu IMS , takich jak ROLB , do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż wsadowe programy DL/I).
 - W systemie IBM należy użyć tego wywołania dla lokalnych jednostek pracy koordynowanych przez menedżera kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) nie może zostać wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek” na stronie 692](#) .
4. Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwyttem kolejki i obejmują takie elementy jak:
- Wartości pól *GroupId*, *MsgSeqNumber*, *Offset* i *MsgFlags* w strukturze MQMD.
 - Określa, czy komunikat jest częścią jednostki pracy.
 - W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.
- Menedżer kolejek przechowuje *trzy* zestawy informacji o grupach i segmentach, jeden zestaw dla każdego z następujących elementów:
- Ostatnie pomyślne wywołanie MQPUT (może to być część jednostki pracy).
 - Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
 - Ostatnie pomyślne wywołanie MQGET, które przeglądało komunikat w kolejce (ten *nie może* być częścią jednostki pracy).
5. Informacje powiązane z wywołaniem MQGET są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zakresem jednostki pracy, nie mają informacji o grupach i segmentach, które zostały odtworzone, jeśli zostanie utworzona kopia zapasowa jednostki pracy.

Odtwarzanie informacji o grupach i segmentach do jej poprzedniej wartości, gdy tworzona jest kopia zapasowa jednostki pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy, a także zrestartowanie w poprawnym punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się.

Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczające informacje, aby móc restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu.

Szczegółowe informacje na temat sposobu restartowania w poprawnym punkcie po awarii systemu można znaleźć w sekcji MQPMO_LOGICAL_ORDER opisanej w sekcji [“MQPMO-Put-message, opcje” na stronie 500](#) oraz w opcji MQGMO_LOGICAL_ORDER opisanej w sekcji [“MQGMO-Opcje Get-message” na stronie 367](#).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy.

6. Jednostka pracy ma ten sam zasięg co uchwyt połączenia. Wszystkie wywołania produktu MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Więcej informacji na temat zasięgu uchwytów połączeń zawiera opis parametru *Hconn* opisanego w sekcji [“MQCONN-Połączenie menedżera kolejek” na stronie 668](#) .
7. To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.
8. Długo działająca aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapętnić kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć tę możliwość, administrator musi ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapętnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duże, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Wywołanie C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQQLONG   CompCode;   /* Completion code */
MQQLONG   Reason;     /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```


Zadeklaruj parametry w następujący sposób:

```
HCONN      DS  F  Connection handle
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE
```

Wywołanie języka Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQBEGIN-Rozpoczęcie jednostki pracy

Wywołanie MQBEGIN rozpoczyna jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zewnętrzne menedżery zasobów.

Składnia

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

Hconn musi być niewspółużytkowanym uchwytem połączenia. Jeśli określono uchwyt połączenia współużytkowanego, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_HCONN_ERROR. Więcej informacji na temat współużytkowanych i niewspółużytkowanych uchwytów można znaleźć w opisie opcji MQCNO_HANDLE_SHARE_* w podręczniku [“MQCNO-opcje połączenia”](#) na stronie 320 .

BeginOptions

Typ: MQBO-input/output

Są to opcje sterujące działaniem komendy MQBEGIN, zgodnie z opisem w sekcji [“MQBO-opcje rozpoczęcia”](#) na stronie 281.

Jeśli nie są wymagane żadne opcje, programy napisane w języku C lub S/390 assembler mogą określać pusty adres parametru, zamiast określać adres struktury MQBO.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_NO_EXTERNAL_UCZESTNICZY

(2121, X'849 ') Nie zarejestrowano żadnych uczestniczących menedżerów zasobów.

MQRC_W_IPANT_NOT_AVAILABLE

(2122, X'84A') Uczestniczy menedżer zasobów nie jest dostępny.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

BŁĄD MQRC_BO_ERROR

(2134, X'856 ') Struktura opcji Begin-options jest niepoprawna.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

Błąd MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwył połączenia nie jest poprawny.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_IN_PROGRESS

(2128, X'850 ') Jednostka pracy już została uruchomiona.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Użyj wywołania MQBEGIN, aby uruchomić jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zmiany w zasobach należących do innych menedżerów zasobów. Menedżer kolejek obsługuje trzy typy jednostek pracy:
 - **Lokalna jednostka pracy koordynowana przez menedżera kolejek:** jednostka pracy, w której menedżer kolejek jest jedynym uczestniczącym menedżerem zasobów, a więc menedżer kolejek działa jako koordynator jednostki pracy.
 - Aby uruchomić ten typ jednostki pracy, należy określić opcję MQPMO_SYNCPOINT lub MQGMO_SYNCPOINT w pierwszej wywołaniu MQPUT, MQPUT1 lub MQGET w jednostce pracy.

- Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołania MQCMIT lub MQBACK.
 - **Menedżer kolejek-koordynowana globalna jednostka pracy:** jednostka pracy, w której menedżer kolejek działa jako koordynator jednostki pracy, zarówno w przypadku zasobów MQ , jak i dla zasobów należących do innych menedżerów zasobów. Te menedżery zasobów współpracują z menedżerem kolejek w celu zapewnienia, że wszystkie zmiany w zasobach w jednostce pracy są zatwierdzane lub wycofane razem.
 - Aby uruchomić ten typ jednostki pracy, należy użyć wywołania MQBEGIN.
 - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań MQCMIT i MQBACK.
 - **Zewnętrznie-koordynowana globalna jednostka pracy:** jednostka pracy, w której menedżer kolejek jest uczestnikiem, ale menedżer kolejek nie działa jako koordynator jednostki pracy. Zamiast tego istnieje zewnętrzny koordynator jednostki pracy, z którym współpracuje menedżer kolejek.
 - Aby rozpocząć ten typ jednostki pracy, należy skorzystać z odpowiedniego wywołania udostępnionego przez zewnętrznego koordynatora jednostki pracy.
 Jeśli wywołanie komendy MQBEGIN jest używane do próby uruchomienia jednostki pracy, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_ENVIRONMENT_ERROR.
 - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań zatwierdzania i tworzenia kopii zapasowych dostarczonych przez zewnętrznego koordynatora jednostki pracy.
 Jeśli do zatwierdzenia lub wycofania jednostki pracy używana jest wywołanie MQCMIT lub MQBACK, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ENVIRONMENT_ERROR.
2. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek” na stronie 692](#) .
 3. Aplikacja może w danym momencie uczestniczyć tylko w jednej jednostce pracy. Wywołanie funkcji MQBEGIN kończy się niepowodzeniem z kodem przyczyny MQRC_UOW_IN_PROGRESS, jeśli istnieje już jednostka pracy dla aplikacji, niezależnie od tego, jaki typ jednostki pracy jest taki sam.
 4. Wywołanie MQBEGIN nie jest poprawne w środowisku klienta MQI produktu MQ . Próba użycia wywołania nie powiodła się. Kod przyczyny: MQRC_ENVIRONMENT_ERROR.
 5. Gdy menedżer kolejek działa jako koordynator jednostki pracy dla globalnych jednostek pracy, menedżerowie zasobów, którzy mogą uczestniczyć w jednostce pracy, są zdefiniowani w pliku konfiguracyjnym menedżera kolejek.
 6. W systemie IBM następujące trzy typy jednostek pracy są obsługiwane w następujący sposób:
 - **Lokalna jednostka pracy koordynowana przez menedżer kolejek** może być używana tylko wtedy, gdy definicja kontroli transakcji nie istnieje na poziomie zadania, tj. komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) nie może zostać wydana dla zadania.
 - Opcja **Menedżer kolejek-koordynowana globalna jednostka pracy** nie jest obsługiwana.
 - **Zewnętrznie-skoordynowana globalna jednostka pracy** może być używana tylko wtedy, gdy definicja kontroli transakcji istnieje na poziomie zadania, tj. komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) musi zostać wydana dla zadania. Jeśli ta operacja została wykonana, operacje IBM i COMMIT i ROLLBACK mają zastosowanie do zasobów MQ , a także do zasobów należących do innych uczestniczących menedżerów zasobów.

Wywołanie C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn; /* Connection handle */
```

```
MQBO      BeginOptions; /* Options that control the action of MQBEGIN */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Options that control the action of MQBEGIN
01 BEGINOPTIONS.
   COPY CMQBOV.
** Completion code
01 COMPCODE       PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON         PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl BeginOptions   like MQBO;     /* Options that control the action of
                                   MQBEGIN */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie języka Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn          As Long 'Connection handle'
Dim BeginOptions   As MQBO 'Options that control the action of MQBEGIN'
Dim CompCode       As Long 'Completion code'
Dim Reason         As Long 'Reason code qualifying CompCode'
```

MQBUFMH-przekształcanie buforu w uchwyt komunikatu

Wywołanie funkcji MQBUFMH przekształca bufor w uchwyt komunikatu i jest odwrotnym wywołaniem wywołania MQMHBUF.

To wywołanie pobiera deskryptor komunikatu i właściwości MQRFH2 w buforze i udostępnia je za pomocą uchwytu komunikatu. Właściwości MQRFH2 w danych komunikatu są, opcjonalnie, usuwane. Pola *Encoding*, *CodedCharSetId* i *Format* w deskrytorze komunikatu są aktualizowane, jeśli jest to konieczne, aby poprawnie opisać zawartość buforu po usunięciu właściwości.

Składnia

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwyttem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, w wątku przekształcaniu buforu w uchwyt komunikatu należy ustanowić poprawne połączenie. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie komendy MQRC_CONNECTION_BROKEN nie powiedzie się.

Hmsg

Typ: MQHMQSG-wejście

Jest to uchwyt komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

HOptyBufMsg

Typ: MQBMHO-wejście

Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki uchwyty komunikatów są generowane z buforów.

Szczegółowe informacje na ten temat zawiera sekcja [“MQBMHO-Opcje uchwytu buforu do obsługi komunikatów”](#) na stronie 279.

MsgDesc

Typ: MQMD-input/output

Struktura *MsgDesc* zawiera właściwości deskryptora komunikatu i opisuje zawartość obszaru buforu.

W przypadku wyjścia z wywołania właściwości są opcjonalnie usuwane z obszaru buforu, a w tym przypadku deskryptor komunikatu jest aktualizowany w celu poprawnego opisanie obszaru buforu.

Dane w tej strukturze muszą znajdować się w zestawie znaków i kodowaniu aplikacji.

BufferLength

Typ: MQLONG-wejście

BufferLength to długość obszaru buforu (w bajtach).

Wartość *BufferLength* równa zero bajtów jest poprawna i wskazuje, że obszar buforu nie zawiera żadnych danych.

Buforuj

Typ: MQBYTEExBufferLength-input/output

Są to opcje sterujące działaniem komendy MQBEGIN, zgodnie z opisem w sekcji [“MQBEGIN-Rozpoczęcie jednostki pracy”](#) na stronie 637.

Buffer definiuje obszar zawierający bufor komunikatów. W przypadku większości danych powinien zostać wyrównany bufor na granicy 4-bajtowej.

Jeśli *Buffer* zawiera dane znakowe lub numeryczne, ustaw wartości pól *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* na wartości odpowiednie dla danych. Umożliwia to przekształcanie danych, jeśli to konieczne.

Jeśli właściwości znajdują się w buforze komunikatów, są one opcjonalnie usuwane, a później stają się dostępne z uchwytu komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *BufferLength* ma wartość zero, *Buffer* nie jest przywołany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

DataLength

Typ: MQLONG-wyjście

Długość (w bajtach) buforu, który może mieć usunięte właściwości.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

BŁĄD MQRC_BMHO_ERROR

(2489, X'09B9') Struktura opcji bufora dla uchwytu komunikatu nie jest poprawna.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'07D4') Parametr buforu nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

Błąd MQRC_MD_ERROR

(2026, X'07EA') deskryptor komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_RFH_ERROR

(2334, X'091E') Struktura MQRFH2 nie jest poprawna.

Błąd formatu MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

Wywołania MQBUFMH nie mogą zostać przechwycone przez wyjścia funkcji API-bufor jest przekształcany w uchwyt komunikatu w obszarze aplikacji; wywołanie nie dociera do menedżera kolejek.

Wywołanie C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,  
         &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;          /* Connection handle */  
MQHMSG  Hmsg;           /* Message handle */  
MQBMHO  BufMsgHOpts;    /* Options that control the action of MQBUFMH */  
MQMD     MsgDesc;       /* Message descriptor */  
MQLONG   BufferLength;   /* Length in bytes of the Buffer area */  
MQBYTE   Buffer[n];     /* Area to contain the message buffer */  
MQLONG   DataLength;    /* Length of the output buffer */  
MQLONG   CompCode;      /* Completion code */  
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,  
                    BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Message handle  
01 HMSG           PIC S9(18) BINARY.  
** Options that control the action of MQBUFMH  
01 BUFMSGHOPTS.  
   COPY CMQBMHOV.  
** Message descriptor  
01 MSGDESC.  
   COPY CMQMD.  
** Length in bytes of the Buffer area  
01 BUFFERLENGTH  PIC S9(9) BINARY.  
** Area to contain the message buffer  
01 BUFFER        PIC X(n).  
** Length of the output buffer  
01 DATALENGTH   PIC S9(9) BINARY.  
** Completion code  
01 COMPCODE      PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON        PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,  
             DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl BufMsgHOpts like MQBMHO;  /* Options that control the action of
                                MQBUFMH */
dcl MsgDesc    like MQMD;     /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer      char(n);       /* Area to contain the message buffer */
dcl DataLength  fixed bin(31); /* Length of the output buffer */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

Wywołanie High Level Assembler

```

CALL MQBUFMH, (HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
              DATALENGTH,COMPCODE,REASON)

```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
BUFMSGHOPTS	CMQBMHOA	,	Options that control the action of MQBUFMH
MSGDESC	CMQMDA	,	Message descriptor
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the properties
DATALENGTH	DS	F	Length of the output buffer
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCB-zarządzanie wywołaniem zwrotnym

Wywołanie MQCB rejestruje wywołanie zwrotne dla podanego uchwytu obiektu i steruje aktywacją i zmianami w wywołaniu zwrotnym.

Wywołanie zwrotne jest to fragment kodu (określony jako nazwa funkcji, która może być dynamicznie dowiązana lub jako wskaźnik funkcji), która jest wywoływana przez składnik IBM MQ po wystąpieniu określonych zdarzeń.

Aby użyć komendy MQCB i MQCTL na kliencie, należy połączyć się z serwerem, na którym wynegocjowany parametr **SHARECNV** kanału uzgodnił wartość niezerową.

Typy wywołań zwrotnych, które mogą być zdefiniowane, to:

Konsument komunikatu

Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat, spełniający określone kryteria wyboru, jest dostępny na uchwycie obiektu.

Tylko jedna funkcja zwrotna może być zarejestrowana dla każdego uchwytu obiektu. Jeśli pojedyncza kolejka ma być odczytywana z wieloma kryteriami wyboru, kolejka musi być otwierana wiele razy, a funkcja konsumenta zarejestrowana na każdym uchwycie.

procedura obsługi zdarzeń

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko wywołań zwrotnych.

Funkcja jest wywoływana, gdy wystąpi warunek zdarzenia, na przykład menedżer kolejek lub połączenie zatrzymujące się lub wyciszające.

Funkcja nie jest wywoływana w przypadku warunków, które są specyficzne dla pojedynczego konsumenta komunikatów, na przykład MQRC_GET_INHIBITED; jest wywoływana, jeśli funkcja zwrotna nie kończy się normalnie.

Składnia

MQCB (*Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts, CompCode, Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można określić następującą wartość specjalną dla *MQHC_DEF_HCONN* , aby użyć uchwytu połączenia powiązanego z tą jednostką wykonywania.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

// operacji

Typ: MQLONG-wejście

Operacja jest przetwarzana dla wywołania zwrotnego zdefiniowanego dla podanego uchwytu obiektu. Należy określić jedną z następujących opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

MQOP_REGISTER

Zdefiniuj funkcję zwrotną dla podanego uchwytu obiektu. Ta operacja definiuje funkcję, która ma zostać wywołana, oraz kryteria wyboru, które mają być używane.

Jeśli funkcja zwrotna jest już zdefiniowana dla uchwytu obiektu, definicja jest zastępowana. Jeśli podczas zastępowania wywołania zwrotnego zostanie wykryty błąd, funkcja jest wyrejestrowana.

Jeśli wywołanie zwrotne jest zarejestrowane w tej samej funkcji zwrotnej, w której została wcześniej wyrejestrowana, jest ona traktowana jako operacja zastępowania. W przypadku wywołań początkowych lub końcowych nie są wywoływane.

Za pomocą komendy MQOP_REGISTER można użyć komendy MQOP_SUSPEND lub MQOP_RESUME.

MQOP_DEREGISTER

Zatrzymaj konsumowanie komunikatów dla uchwytu obiektu i usuwa uchwyt z tych zakwalifikowanych do wywołania zwrotnego.

Wywołanie zwrotne jest automatycznie wyrejestrowywane, jeśli powiązany uchwyt jest zamknięty.

Jeśli wywołanie MQOP_DEREGISTER zostanie wywołane z poziomu konsumenta, a wywołanie zwrotne ma zdefiniowane wywołanie zatrzymania, zostanie ono wywołane po powrocie ze strony konsumenta.

Jeśli ta operacja zostanie wywołana dla *Hobj* bez zarejestrowanego konsumenta, wywołanie zwraca wartość MQRC_CALLBACK_NOT_REGISTERED.

MQOP_SUSPEND

Zawiesza korzystanie z komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie otrzymała zdarzeń podczas zawieszania, a wszystkie zdarzenia, które nie zostały pominięte w stanie zawieszenia, nie zostaną udostępnione operacji po jej wznowieniu.

Po zawieszeniu funkcja konsumenta kontynuuje wywoływanie zwrotnych typów sterowania.

MQOP_RESUME

Wznów korzystanie z komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie otrzymała zdarzeń podczas zawieszania, a wszystkie zdarzenia, które nie zostały pominięte w stanie zawieszenia, nie zostaną udostępnione operacji po jej wznowieniu.

CallbackDesc

Typ: MQCBD-wejście

Jest to struktura identyfikująca funkcję zwrotną, która jest rejestrowana przez aplikację, oraz opcje używane podczas rejestrowania.

Szczegółowe informacje na temat struktury zawiera sekcja [MQCBD](#).

Deskryptor wywołania zwrotnego jest wymagany tylko w przypadku opcji MQOP_REGISTER. Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Hobj

Typ: MQHOBJ-wejście

Ten uchwyt reprezentuje dostęp, który został utworzony dla obiektu, z którego komunikat ma zostać zużyty. Jest to uchwyt, który został zwrócony z poprzednich wywołań [MQOPEN](#) lub [MQSUB](#) (w parametrze *Hobj*).

Produkt *Hobj* nie jest wymagany podczas definiowania procedury obsługi zdarzeń (MQCBT_EVENT_HANDLER) i należy ją określić jako MQHO_NONE.

Jeśli produkt *Hobj* został zwrócony z wywołania MQOPEN, kolejka musi zostać otwarta z jedną lub więcej spośród następujących opcji:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Typ: MQMD-dane wejściowe

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu.

Parametr *MsgDesc* definiuje atrybuty komunikatów wymaganych przez konsumenta, a wersja deskryptora MQMD, która ma zostać przekazana do konsumenta komunikatów.

Opcje *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* i *Offset* w strukturze MQMD są używane do wyboru komunikatów, w zależności od opcji określonych w parametrze *GetMsgOpts*.

Opcje *Encoding* i *CodedCharSetId* są używane do konwersji komunikatów, jeśli zostanie określona opcja MQGMO_CONVERT.

Szczegółowe informacje na ten temat zawiera sekcja [MQMD](#).

Produkt *MsgDesc* jest używany dla zmiennej MQOP_REGISTER, a jeśli wymagane są wartości inne niż domyślne dla wszystkich pól. Program *MsgDesc* nie jest używany dla procedury obsługi zdarzeń.

Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Należy zauważyć, że jeśli wielu konsumentów jest zarejestrowanych w tej samej kolejce z nakładającymi się selektorami, wybrany konsument dla każdego komunikatu jest niezdefiniowany.

GetMsgOpts (GetMsg)

Typ: MQGMO-wejście

Parametr *GetMsgOpts* określa sposób, w jaki konsument komunikatów pobiera komunikaty. Wszystkie opcje tego parametru mają znaczenie w sposób opisany w sekcji "[MQGMO-Opcje GetMessage](#)" na stronie 367, jeśli są używane w wywołaniu MQGET, z wyjątkiem:

MQGMO_SET_SIGNAL

Ta opcja nie jest dozwolona.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_*

Kolejność komunikatów dostarczanych do konsumenta przeglądania jest podyktowana kombinacjami tych opcji. Istotne kombinacje to:

MQGMO_BROWSE_FIRST

Pierwszy komunikat w kolejce jest dostarczany wielokrotnie do konsumenta. Jest to przydatne w sytuacji, gdy konsument destrukcyjnie konsumuje komunikat w wywołaniu zwrotnym. Tej opcji należy używać z ostrożnością.

MQGMO_BROWSE_NEXT

Konsument otrzymuje każdy komunikat w kolejce, od bieżącej pozycji kursora do momentu osiągnięcia końca kolejki.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT

Kursor zostanie zresetowany do początku kolejki. Następnie konsumentowi podaje się każdy komunikat do momentu, gdy kursor osiągnie koniec kolejki.

MQGMO_BROWSE_FIRST + MQGMO_MARK_*

Rozpoczynając od początku kolejki, konsument otrzymuje pierwszą niezaznaczoną wiadomość w kolejce, która jest następnie oznaczona dla tego konsumenta. Ta kombinacja zapewnia, że konsument może odbierać nowe komunikaty dodane za bieżącym punktem kursora.

MQGMO_BROWSE_NEXT + MQGMO_MARK_*

Począwszy od pozycji kursora, konsument otrzymuje następną niezaznaczoną wiadomość w kolejce, która jest następnie oznaczana dla tego konsumenta. Tej kombinacji należy używać z ostrożnością, ponieważ komunikaty mogą być dodawane do kolejki za bieżącą pozycją kursora.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_*

Ta kombinacja nie jest dozwolona. Jeśli używane jest wywołanie, zwraca wartość MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT i WaitInterval

Te opcje sterują sposobem wywoływania konsumenta.

MQGMO_NO_WAIT

Konsument nie jest nigdy wywoływany z parametrem MQRC_NO_MSG_AVAILABLE. Konsument jest wywoływany tylko w przypadku komunikatów i zdarzeń.

MQGMO_WAIT z zerem WaitInterval

Kod MQRC_NO_MSG_AVAILABLE jest przekazywany do konsumenta, gdy nie są dostępne żadne komunikaty, a konsument został uruchomiony lub konsument został dostarczony co najmniej jeden komunikat od ostatniego kodu przyczyny "brak komunikatów".

Uniemożliwia to konsumentowi odpytywanie w pętli zajętości, gdy określony jest przedział czasu oczekiwania na zero.

MQGMO_WAIT i dodatnia WaitInterval

Konsument jest wywoływany po upływie określonego przedziału czasu oczekiwania z kodem przyczyny MQRC_NO_MSG_AVAILABLE. Wywołanie to jest wykonywane niezależnie od tego, czy do konsumenta dostarczono jakiegokolwiek komunikaty. Pozwala to użytkownikowi na przetwarzanie pulsu lub przetwarzania typu wsadowego.

MQGMO_WAIT i WaitInterval z tabeli MQWI_UNLIMITED

Określa on nieskończone oczekiwanie przed zwróceniem wartości MQRC_NO_MSG_AVAILABLE. Konsument nie jest nigdy wywoływany z parametrem MQRC_NO_MSG_AVAILABLE.

Produkt *GetMsgOpts* jest używany tylko w przypadku komendy MQOP_REGISTER i jeśli wymagane są wartości inne niż domyślne dla wszystkich pól. Program *GetMsgOpts* nie jest używany dla procedury obsługi zdarzeń.

Jeśli parametr *GetMsgOpts* nie jest wymagany, przekazany adres parametru może mieć wartość NULL. Użycie tego parametru jest takie samo, jak podanie parametru MQGMO_DEFAULT razem z opcją MQGMO_FAIL_IF_QUIESCING.

Jeśli uchwyt właściwości komunikatu jest udostępniany w strukturze MQGMO, kopia jest udostępniana w strukturze MQGMO, która jest przekazywana do wywołania zwrotnego konsumenta. Po powrocie z wywołania MQCB aplikacja może usunąć uchwyt właściwości komunikatu.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kody przyczyny znajdujące się na poniższej liście to te, które menedżer kolejek może zwrócić dla parametru *Reason*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

Błąd MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') Niepoprawne pole typu wywołania zwrotnego.

MQRC_CALLBACK_NOT_ZAREJESTROWANY

(2448, X' 990 ') Nie można wyrejestrować, zawiesić lub wznowić, ponieważ nie ma zarejestrowanej procedury zwrotnej.

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') Należy podać wartość *CallbackFunction* lub *CallbackName*, ale nie obie jednocześnie.

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') Niepoprawne pole typu wywołania zwrotnego.

MQRC_CBD_OPTIONS_ERROR

(2484, X'9B4') Niepoprawne pole opcji MQCBD.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_CORREL_ID_ERROR (BŁĄD)

(2207, X'89F') Błąd korelacji identyfikatora.

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') Parametr długości danych nie jest poprawny.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_GET_INHIBITED

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

BŁĄD MQRC_GMO_ERROR

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

MQRC_HANDLE_IN_USE_DLA_UOW

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

BŁĄD MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Opcje zgodności nie są poprawne.

MQRC_MAX_MSG_LENGTH_ERROR

(2485, X'9B4') Niepoprawne pole *MaxMsgLength* .

Błąd MQRC_MD_ERROR

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') Określony punkt wejścia funkcji nie został znaleziony w module.

Niepoprawna wartość MQRC_MODULE_INVALID

(2496, X'9C0') Znaleziono moduł, jednak jest to niepoprawny typ; nie jest to 32-bitowa, 64-bitowa lub poprawna biblioteka dołączana dynamicznie.

MQRC_MODULE_NOT_FOUND

(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie został autoryzowany do załadowania.

MQRC_MSG_SEQ_NUMBER_ERROR,

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') Użycie znacznika komunikatu nie jest poprawne.

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') Brak dostępnego komunikatu.

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') Przeglądaj kursor nie umieszczony na komunikacie.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

BŁĄD MQRC_OPERATION_ERROR

(2206, X'89E') Niepoprawny kod operacji w wywołaniu API Call.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815 ') Signal outstanding for this handle.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') Obsługa punktów synchronizacji nie jest dostępna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Baza MQCB jest używana do definiowania działania, które ma być wywoływane dla każdego komunikatu, zgodnego z podanymi kryteriami, dostępnymi w kolejce. Po przetworzeniu działania komunikat jest usuwany z kolejki i przekazywany do zdefiniowanego konsumenta komunikatów lub udostępniany jest znacznik komunikatu, który jest używany do pobierania komunikatu.
2. Obiekt MQCB może być używany do definiowania procedur zwrotnych przed rozpoczęciem korzystania z komendy MQCTL lub z poziomu procedury zwrotnej.
3. Aby użyć obiektu MQCB z poziomu poza procedurą wywołania zwrotnego, należy najpierw zawiesić wykorzystanie komunikatów za pomocą komendy MQCTL i wznowić korzystanie z niej.
4. Baza MQCB nie jest obsługiwana w ramach adaptera IMS.

Sekwencja wywołań zwrotnych konsumenta komunikatów

Istnieje możliwość skonfigurowania konsumenta w taki sposób, aby wywoływało wywołanie zwrotne w kluczowych punktach podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- gdy połączenie jest uruchomione,
- gdy połączenie jest zatrzymane i
- gdy konsument jest wyrejestrowany, jawnie lub niejawnie przez operację MQCLOSE.

Tabela 101. Definicje komend MQCTL	
Czasownik	Znaczenie
MQCTL (POCZĄTEK)	Wywołanie MQCTL przy użyciu operacji MQOP_START
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji MQOP_STOP
MQCTL (WAIT)	Wywołanie MQCTL przy użyciu operacji MQOP_START_WAIT

Dzięki temu konsument może zachować stan powiązany z konsumentem. Jeśli aplikacja zażądała wywołania zwrotnego, reguły wywołania konsumenta są następujące:

Zarejestruj

Jest zawsze pierwszym typem wywołania zwrotnego.

Jest zawsze wywoływany w tym samym wątku, co wywołanie MQCB (REGISTER).

START

Jest zawsze wywoływana synchronicznie za pomocą komendy MQCTL (START).

- Wszystkie wywołania zwrotne START są zakończone przed zwrotami komendy MQCTL (START).

Znajduje się w tym samym wątku, co dostarczanie komunikatu, jeśli zażądano THREAD_AFFINITY.

Wywołanie z uruchomieniem nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne MQCTL (STOP) podczas operacji MQCTL (START) zostanie uruchomione.

STOP

Po wywołaniu tego połączenia nie zostaną dostarczone żadne dodatkowe komunikaty ani żadne zdarzenia, dopóki połączenie nie zostanie zrestartowane.

Wartość STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana na potrzeby START, lub komunikat lub zdarzenie.

DEREGISTER

Jest zawsze ostatnim typem wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie oparte na wątkach w wywołaniach zwrotnych START i STOP. Inicjowanie i czyszczenie oparte na wątkach można wykonywać za pomocą wywołań zwrotnych REGISTER i DEREGISTER.

Nie należy podejmować żadnych założeń dotyczących życia i dostępności wątku innego niż to, co jest określone. Na przykład, nie należy polegać na wątku pozostający przy życiu poza ostatnim wywołaniem DEREGISTER. Podobnie, jeśli wybrano opcję nieużywania powinowactwa THREAD_AFFINITY, nie należy zakładać, że wątek istnieje za każdym razem, gdy połączenie jest uruchomione.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątku, może zawsze utworzyć wątek, a następnie użyć komendy MQCTL (WAIT). Ma to wpływ na "oddawanie" wątku do IBM MQ na potrzeby asynchronicznego dostarczania komunikatów.

Użycie połączenia konsumenta komunikatów

Istnieje możliwość skonfigurowania konsumenta w taki sposób, aby wywoływało wywołanie zwrotne w kluczowych punktach podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- gdy połączenie jest uruchomione,
- gdy połączenie jest zatrzymane i
- gdy konsument jest wyrejestrowany, jawnie lub niejawnie przez operację MQCLOSE.

Tabela 102. Definicje komend MQCTL	
Czasownik	Znaczenie
MQCTL (POCZĄTEK)	Wywołanie MQCTL przy użyciu operacji MQOP_START
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji MQOP_STOP
MQCTL (WAIT)	Wywołanie MQCTL przy użyciu operacji MQOP_START_WAIT

Dzięki temu konsument może zachować stan powiązany z konsumentem. Jeśli aplikacja zażądała wywołania zwrotnego, reguły wywołania konsumenta są następujące:

Zarejestruj

Jest zawsze pierwszym typem wywołania zwrotnego.

Jest zawsze wywoływany w tym samym wątku, co wywołanie MQCB (REGISTER).

START

Jest zawsze wywoływana synchronicznie za pomocą komendy MQCTL (START).

- Wszystkie wywołania zwrotne START są zakończone przed zwrotami komendy MQCTL (START).

Znajduje się w tym samym wątku, co dostarczanie komunikatu, jeśli zażądano THREAD_AFFINITY.

Wywołanie z uruchomieniem nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne MQCTL (STOP) podczas operacji MQCTL (START) zostanie uruchomione.

STOP

Po wywołaniu tego połączenia nie zostaną dostarczone żadne dodatkowe komunikaty ani żadne zdarzenia, dopóki połączenie nie zostanie zrestartowane.

Wartość STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana na potrzeby START, lub komunikat lub zdarzenie.

DEREGISTER

Jest zawsze ostatnim typem wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie oparte na wątkach w wywołaniach zwrotnych START i STOP. Inicjowanie i czyszczenie oparte na wątkach można wykonywać za pomocą wywołań zwrotnych REGISTER i DEREGISTER.

Nie należy podejmować żadnych założeń dotyczących życia i dostępności wątku innego niż to, co jest określone. Na przykład, nie należy polegać na wątku pozostający przy życiu poza ostatnim wywołaniem DEREGISTER. Podobnie, jeśli wybrano opcję nieużywania powinowactwa THREAD_AFFINITY, nie należy zakładać, że wątek istnieje za każdym razem, gdy połączenie jest uruchomione.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątku, może zawsze utworzyć wątek, a następnie użyć komendy MQCTL (WAIT). Ma to wpływ na "oddawanie" wątku do IBM MQ na potrzeby asynchronicznego dostarczania komunikatów.

Wywołanie C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,  
GetMsgOpts, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;          /* Connection handle */  
MQLONG   Operation;      /* Operation being processed */  
MQCBD    CallbackDesc;   /* Callback descriptor */  
MQHOBJS  Hobj;           /* Object handle */  
MQMD     MsgDesc         /* Message descriptor attributes */  
MQGMO    GetMsgOpts      /* Message options */  
MQLONG   CompCode;       /* Completion code */  
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJS, MSGDESC,  
GETMSGOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN PIC S9(9) BINARY.  
** Operation  
01 OPERATION PIC S9(9) BINARY.  
** Callback Descriptor  
01 CBDESC.  
COPY CMQCBDEV.  
01 HOBJS PIC S9(9) BINARY.  
** Message Descriptor  
01 MSGDESC.  
COPY CMQMDV.  
** Get Message Options  
01 GETMSGOPTS.  
COPY CMQGMOV.  
** Completion code  
01 COMPCODE PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,  
          CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */  
dcl Operation  fixed bin(31); /* Operation */  
dcl CallbackDesc like MQCBD; /* Callback Descriptor */  
dcl Hobj       fixed bin(31); /* Object Handle */  
dcl MsgDesc    like MQMD; /* Message Descriptor */  
dcl GetMsgOpts like MQGMO; /* Get Message Options */  
dcl CompCode   fixed bin(31); /* Completion code */  
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

MQCB_FUNCTION-funkcja Callback

Wywołanie funkcji MQCB_FUNCTION jest funkcją zwrotną dla obsługi zdarzeń i asynchronicznego wykorzystania komunikatów.

Definicja wywołania MQCB_FUNCTION jest udostępniana wyłącznie w celu opisanie parametrów, które są przekazywane do funkcji zwrotnej. Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQCB_FUNCTION.

Specyfikacja rzeczywistej funkcji, która ma zostać wywołana, jest danymi wejściowymi dla wywołania [MQCB](#) i przekazywana jest za pośrednictwem struktury [MQCBD](#).

Składnia

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następujące wartości określone dla parametru *Hconn*:

MQHC_DEF_CONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

MsgDesc

Typ: MQMD-dane wejściowe

Ta struktura opisuje atrybuty pobranego komunikatu.

Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Przekazana wersja deskryptora MQMD jest taka sama, jak wersja przekazana w wywołaniu MQCB, która zdefiniował funkcję konsumenta.

Adres deskryptora MQMD jest przekazywany jako znak o wartości NULL, jeśli do żądania zwrócenia uchwytu komunikatu zamiast deskryptora MQMD użyto wersji 4 MQGMO.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

GetMsgOpts (GetMsg)

Typ: MQGMO-wejście

Opcje służące do sterowania działaniami konsumenta komunikatów. Ten parametr zawiera również dodatkowe informacje na temat zwróconego komunikatu.

Szczegółowe informacje na ten temat zawiera sekcja [MQGMO](#).

Przekazana wersja produktu MQGMO to najnowsza obsługiwana wersja.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

Buforuj

Typ: MQBYTEExBufferDługość-wejście

Jest to obszar zawierający dane komunikatu.

Jeśli dla tego wywołania nie jest dostępny żaden komunikat lub jeśli komunikat nie zawiera danych komunikatu, adres serwera *Buffer* jest przekazywany jako wartość NULL.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

Kontekst

Typ: MQCBC-input/output

Ta struktura udostępnia informacje kontekstowe do funkcji zwrotnych. Szczegółowe informacje na ten temat zawiera sekcja [“MQCBC-kontekst wywołania zwrotnego”](#) na stronie 283.

Użycie notatek

1. Należy pamiętać, że jeśli procedury zwrotne korzystają z usług, które mogą opóźnić lub zablokować wątek, na przykład MQGET z oczekiwaniem, może opóźnić wysyłkę innych wywołań zwrotnych.
2. Oddzielna jednostka pracy nie jest automatycznie ustanawiana dla każdego wywołania procedury zwrotnej, dlatego procedury mogą wydać wywołanie zatwierdzenia lub odroczyć zatwierdzenie do czasu przetworzenia logicznej zadania wsadowego pracy. Gdy zadanie wsadowe pracy jest zatwierdzane, zatwierdza komunikaty dla wszystkich funkcji wywołania zwrotnego, które zostały wywołane od ostatniego punktu synchronizacji.
3. Programy wywoływane przez program CICS LINK lub CICS START pobierają parametry za pomocą usług CICS przy użyciu nazwanych obiektów znanych jako kontenery kanałów. Nazwy kontenerów są takie same, jak nazwy parametrów. Więcej informacji na ten temat zawiera dokumentacja produktu CICS.
4. Procedury wywołania zwrotnego mogą wywoływać wywołanie MQDISC, ale nie dla ich własnego połączenia. Na przykład, jeśli procedura zwrotna utworzyła połączenie, może ona również odłączyć połączenie.
5. Procedura zwrotna nie powinna generalnie polegać na tym, że za każdym razem jest wywoływana z tego samego wątku. Jeśli jest to wymagane, należy użyć powinowactwa MQCTLO_THREAD_AFFINITY podczas uruchamiania połączenia.
6. Gdy procedura zwrotna otrzymuje niezerowy kod przyczyny, musi podjąć odpowiednie działanie.
7. Funkcja MQCB_FUNCTION nie jest obsługiwana w adapterze IMS.

MQCLOSE-zamknięcie obiektu

Wywołanie MQCLOSE zrzekło się dostępu do obiektu i jest odwrotną wersją wywołań MQOPEN i MQSUB.

Składnia

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN i podać następującą wartość dla produktu *Hconn*:

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Hobj

Typ: MQHOBJ-input/output

Ten uchwyt reprezentuje obiekt, który jest zamykany. Obiekt może być dowolnego typu. Wartość *Hobj* została zwrócona przez poprzednie wywołanie MQOPEN.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia ten parametr na wartość, która nie jest poprawnym uchwytami dla środowiska. Ta wartość jest następująca:

MQHO_UNUSABLE_HOBJ

Uchwyt obiektu nie do użycia.

W systemie z/OS wartość *Hobj* jest ustawiana na wartość, która jest niezdefiniowana.

Opcje

Typ: MQLONG-wejście

Ten parametr określa sposób zamykania obiektu.

Tylko trwałe kolejki dynamiczne i subskrypcje mogą być zamykane w więcej niż jeden sposób, ponieważ muszą być zachowywane lub usuwane. Są to kolejki z atrybutem *DefinitionType* o wartości MQQDT_PERMANENT_DYNAMIC (patrz atrybut *DefinitionType* opisany w sekcji [“Atrybuty dla kolejek” na stronie 845](#)). W tym temacie podsumowane są opcje zamknięcia.

Trwałe subskrypcje mogą być przechowywane lub usuwane. Te subskrypcje są tworzone przy użyciu wywołania MQSUB z opcją MQSO_DURABLE.

Podczas zamykania uchwytu do zarządzanego miejsca docelowego (jest to parametr *Hobj* zwrócony w wywołaniu MQSUB, w którym użyto opcji MQSO_MANAGED) menedżer kolejek czyści wszystkie publikacje, które nie zostały pobrane, gdy powiązana subskrypcja również została usunięta. Subskrypcja jest usuwana przy użyciu opcji MQCO_REMOVE_SUB w parametrze *Hsub* zwróconej w wywołaniu MQSUB. Uwaga: MQCO_REMOVE_SUB jest domyślnym zachowaniem w tabeli MQCLOSE dla nietrwałej subskrypcji.

Podczas zamykania uchwytu do niezarządzanego miejsca docelowego, użytkownik jest odpowiedzialny za czyszczenie kolejki, w której wysyłane są publikacje. Zamknij subskrypcję za pomocą komendy MQCO_REMOVE_SUB, a następnie przetwórz komunikaty z kolejki do momentu, gdy nie pozostanie żaden.

Należy określić jedną opcję tylko z następujących elementów:

Opcje kolejki dynamicznej: Te opcje kontrolują sposób zamykania trwałych kolejek dynamicznych.

MQCO_DELETE

Kolejka jest usuwana, jeśli spełniony jest jeden z poniższych warunków:

- Jest to stała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i brak komunikatów w kolejce i brak niezatwierdzonych żądań pobierania lub umieszczania żądań dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).

- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło *Hobj*. W takim przypadku wszystkie komunikaty znajdujące się w kolejce są usuwane.

We wszystkich innych przypadkach, w tym w przypadku, gdy produkt *Hobj* został zwrócony w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTION_NOT_VALID_FOR_TYPE, a obiekt nie jest usuwany.

W systemie z/OS, jeśli kolejka jest kolejką dynamiczną, która została logicznie usunięta, a jest to ostatni uchwyt dla niej, kolejka jest fizycznie usuwana. Więcej informacji na ten temat zawiera sekcja "Użycie notatek" na stronie 661.

MQCO_DELETE_PURGE

Kolejka zostanie usunięta, a wszystkie komunikaty na niej usunięte, jeśli spełniony jest jeden z poniższych warunków:

- Jest to stała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i nie ma żadnych niezatwierdzonych żądań pobrania lub umieszczenia oczekujących żądań dla kolejki (dla bieżącego zadania lub innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło *Hobj*.

We wszystkich innych przypadkach, w tym w przypadku, gdy produkt *Hobj* został zwrócony w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTION_NOT_VALID_FOR_TYPE, a obiekt nie jest usuwany.

Tabela pokazuje, które opcje zamknięcia są poprawne oraz czy obiekt jest zachowywany, czy usuwany.			
Typ obiektu lub kolejki	MQCO_NONE	MQCO_DELETE	MQCO_DELETE_PURGE
Obiekt inny niż kolejka	Zachowany	Niepoprawne	Niepoprawne
Kolejka predefiniowana	Zachowany	Niepoprawne	Niepoprawne
stała kolejka dynamiczna	Zachowany	Usunięto, jeśli jest puste i nie ma oczekujących aktualizacji	Komunikaty usunięte; kolejka usunięta, jeśli nie ma oczekujących aktualizacji
Tymczasowa kolejka dynamiczna (wywołanie wydane przez twórcę kolejki)	Usunięte	Usunięte	Usunięte
Tymczasowa kolejka dynamiczna (wywołanie nie zostało wysłane przez twórcę kolejki)	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna	Zachowany	Niepoprawne	Niepoprawne
Miejsce docelowe zarządzanego subskrypcji	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna (subskrypcja została usunięta)	Komunikaty usunięte; kolejka usunięta	Niepoprawne	Niepoprawne

Opcje zamknięcia subskrypcji: Te opcje kontrolują, czy trwałe subskrypcje są usuwane po zamknięciu uchwytu, oraz czy publikacje nadal oczekujące na odczytanie przez aplikację są czyszczone. Te opcje są poprawne tylko do użycia z uchwytym obiektu zwróconego w parametrze *Hsub* wywołania MQSUB.

MQCO_KEEP_SUB

Uchwyt do subskrypcji jest zamknięty, ale subskrypcja jest przechowywana. Publikacje są nadal wysyłane do miejsca docelowego określonego w subskrypcji. Ta opcja jest poprawna tylko wtedy, gdy subskrypcja została wykonana z opcją MQSO_DURABLE.

MQCO_KEEP_SUB jest wartością domyślną, jeśli subskrypcja jest trwała

MQCO_REMOVE_SUB

Subskrypcja zostanie usunięta, a uchwyt do subskrypcji jest zamknięty.

Parametr *Hobj* wywołania MQSUB nie jest unieważniony przez zamknięcie parametru *Hsub* i może być nadal używany dla operacji MQGET lub MQCB w celu odebrania pozostałych publikacji. Jeśli parametr *Hobj* wywołania MQSUB jest również zamknięty, to jeśli jest to zarządzane miejsce docelowe, wszystkie niepobrane publikacje są usuwane.

MQCO_REMOVE_SUB jest wartością domyślną, jeśli subskrypcja nie jest trwała.

Pomyślne zakończenie operacji MQCO_REMOVE_SUB nie oznacza, że działanie zostało zakończone. Aby sprawdzić, czy to wywołanie zostało zakończone, zapoznaj się z krokiem DELETE SUB w sekcji Sprawdzanie, czy komendy asynchroniczne dla rozproszonych sieci zostały zakończone.

Te opcje zamknięcia subskrypcji są podsumowane w poniższych tabelach.

Aby zamknąć uchwyt subskrypcji trwałej, ale zachować subskrypcję, należy użyć następujących opcji zamknięcia subskrypcji:

Zadanie	Opcja zamknięcia subskrypcji
Przechowuj publikacje na uchwycie MQOPENed	MQCO_KEEP_SUB
Usuń publikacje na uchwycie MQOPENed	Działanie jest niedozwolone
Przechowuj publikacje na uchwycie MQSO_MANAGED	MQCO_KEEP_SUB
Usuwanie publikacji z uchwytu MQSO_MANAGED	Działanie jest niedozwolone

Aby anulować subskrypcję, zamykając trwały uchwyt subskrypcji i anulować subskrypcję lub zamknąć uchwyt subskrypcji nietrwałej, użyj następujących opcji zamknięcia subskrypcji:

Zadanie	Opcja zamknięcia subskrypcji
Przechowuj publikacje na uchwycie MQOPENed	MQCO_REMOVE_SUB
Usuń publikacje na uchwycie MQOPENed	Działanie jest niedozwolone
Przechowuj publikacje na uchwycie MQSO_MANAGED	MQCO_REMOVE_SUB

Opcje odczytu z wyprzedzeniem: Następujące opcje sterują tym, co dzieje się z nietrwałymi komunikatami, które zostały wysłane do klienta przed zażądaniem ich przez aplikację i nie zostały jeszcze wykorzystane przez aplikację. Komunikaty te są przechowywane w buforze odczytu z wyprzedzeniem klienta oczekującego na żądanie przez aplikację i mogą zostać usunięte lub skonsumowane z kolejki przed zakończeniem operacji MQCLOSE.

MQCO_IMMEDIATE

Obiekt jest zamykany natychmiast, a wszystkie komunikaty, które zostały wysłane do klienta, zanim aplikacja zażądała ich usunięcia i nie są dostępne do wykorzystania przez żadną aplikację. Jest to wartość domyślna.

MQCO_QUIESCE

Żądanie zamknięcia obiektu jest wykonywane, ale jeśli wszystkie komunikaty, które zostały wysłane do klienta przed zażądaniem ich przez aplikację, nadal znajdują się w buforze odczytu z wyprzedzeniem klienta, wywołanie MQCLOSE zwraca ostrzeżenie MQRC_READ_AHEAD_MSGS i uchwyt obiektu pozostaje poprawny.

Aplikacja może następnie nadal używać uchwytu obiektu do pobierania komunikatów aż do momentu, gdy nie będzie już dostępny, a następnie ponownie zamknąć obiekt. Nie ma więcej wiadomości wysyłanych do klienta przed aplikacją żądającą ich, odczyt z wyprzedzeniem jest teraz wyłączony.

Zaleca się, aby aplikacje używały komendy MQCO_QUIESCE, zamiast próbować dotrzeć do punktu, w którym nie ma więcej komunikatów w buforze odczytu z wyprzedzeniem klienta, ponieważ może dojść do połączenia między ostatnim wywołaniem MQGET i następującą tabelą MQCLOSE, która zostanie odrzucona, jeśli użyto komendy MQCO_IMMEDIATE.

Jeśli komenda MQCLOSE z opcją MQCO_QUIESCE jest wydawana z asynchronicznej funkcji zwrotnej, zastosowanie ma takie samo zachowanie odczytu z wyprzedzeniem. Jeśli zostanie zwrócone ostrzeżenie MQRC_READ_AHEAD_MSGS, to funkcja zwrotna zostanie wywołana co najmniej raz. Gdy ostatni pozostały komunikat, który został odczytany z wyprzedzeniem, został przekazany do funkcji wywołania zwrotnego, pole ConsumerFlags komendy MQCBC jest ustawione na wartość MQCBCF_READA_BUFFER_EMPTY.

Opcja domyślna: Jeśli nie jest wymagana żadna z opisanych wcześniej opcji, można użyć następującej opcji:

MQCO_NONE

Opcjonalne przetwarzanie zamknięcia nie jest wymagane.

Ten *musi* być określony dla:

- Obiekty inne niż kolejki
- Kolejki predefiniowane
- Tymczasowe kolejki dynamiczne (ale tylko w tych przypadkach, w których *Hobj* nie jest to uchwyt zwracany przez wywołanie MQOPEN, które utworzyło kolejkę).
- Lista dystrybucyjna

We wszystkich powyższych przypadkach obiekt jest zachowywany i nie jest usuwany.

Jeśli ta opcja jest określona dla tymczasowej kolejki dynamicznej:

- Kolejka jest usuwana, jeśli została utworzona za pomocą wywołania MQOPEN, które zwróciło *Hobj* ; wszystkie komunikaty, które znajdują się w kolejce, są czyszczone.
- We wszystkich innych przypadkach kolejka (i wszystkie komunikaty) są zachowywane.

Jeśli ta opcja jest określona dla trwałej kolejki dynamicznej, kolejka jest zachowywana i nie została usunięta.

W systemie z/OS, jeśli kolejka jest kolejką dynamiczną, która została logicznie usunięta, a jest to ostatni uchwyt dla niej, kolejka jest fizycznie usuwana. Więcej informacji na ten temat zawiera sekcja ["Użycie notatek"](#) na stronie 661.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Podane kody przyczyn to te, które menedżer kolejek może zwrócić dla parametru *Reason* .

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') Grupa komunikatów nie została zakończona.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') Komunikat logiczny nie został zakończony.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_NOT_AVAILABLE

(2345, X' 929 ') Obiekt sprzęgający nie jest dostępny.

MQRC_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X' 945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE

(2342, X' 926 ') podsystem Db2 nie jest dostępny.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

MQRC_OPTION_NOT_VALID_FOR_TYPE

(2045, X'7FD') W wywołaniu MQOPEN lub MQCLOSE: opcja nie jest poprawna dla typu obiektu.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_NOT_EMPTY

(2055, X'807 ') Kolejka zawiera jeden lub więcej komunikatów lub niezatwierdzonych żądań umieszczania lub pobierania.

MQRC_READ_AHEAD_MSGS

(nnnn, X'xxx ') Klient ma komunikaty odczytu z wyprzedzeniem, które nie zostały jeszcze wykorzystane przez aplikację.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SECURITY_ERROR,

(2063, X'80F') Wystąpił błąd zabezpieczeń.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Gdy aplikacja zgłosi wywołanie MQDISC lub zakończy się normalnie lub nieprawidłowo, wszystkie obiekty, które zostały otwarte przez aplikację i są nadal otwarte, są automatycznie zamykane za pomocą opcji MQCO_NONE.
2. Jeśli zamykany obiekt jest *kolejką*, mają zastosowanie następujące punkty:
 - Jeśli operacje w kolejce wykonywane są jako część jednostki pracy, kolejka może zostać zamknięta przed lub po wystąpieniu punktu synchronizacji bez wpływu na wynik punktu synchronizacji. Jeśli kolejka jest wyzwalana, wykonanie wycofania przed zamknięciem kolejki może spowodować, że komunikat wyzwalacza zostanie wygenerowany. Więcej informacji na temat wyzwalaczy zawiera sekcja [Właściwości komunikatów wyzwalacza](#).
 - Jeśli kolejka została otwarta za pomocą opcji MQOO_BROWSE, kursor przeglądania zostanie zniszczony. Jeśli kolejka jest następnie ponownie otwierana za pomocą opcji MQOO_BROWSE, tworzony jest nowy kursor przeglądania (patrz [MQOO_BROWSE](#)).
 - Jeśli komunikat jest obecnie zablokowany dla tego uchwytu w czasie wywołania MQCLOSE, blokada jest zwalniana (patrz [MQGMO_LOCK](#)).
 - W systemie z/OS, jeśli istnieje żądanie MQGET z opcją MQGMO_SET_SIGNAL, która jest nierozstrzygana względem zamkniętego uchwytu kolejki, żądanie zostanie anulowane (patrz [MQGMO_SET_SIGNAL](#)). Żądania sygnału dla tej samej kolejki, ale złożone dla różnych uchwytów (*Hobj*), nie mają wpływu (chyba, że usuwana jest kolejka dynamiczna, w której to przypadku są również anulowane).
3. Jeśli zamykany obiekt jest *kolejką dynamiczną* (trwałą lub tymczasową), mają zastosowanie następujące punkty:
 - W przypadku kolejki dynamicznej można określić opcje MQCO_DELETE i MQCO_DELETE_PURGE niezależnie od opcji określonych w odpowiednim wywołaniu MQOPEN.

- Po usunięciu kolejki dynamicznej wszystkie wywołania MQGET z opcją MQGMO_WAIT, które są zaległe z kolejką, są anulowane, a kod przyczyny MQRC_Q_DELETED (kod przyczyny) jest zwracany. Patrz MQGMO_WAIT.

Mimo że aplikacje nie mogą uzyskać dostępu do usuniętej kolejki, kolejka nie jest usuwana z systemu, a powiązane zasoby nie są zwalniane, dopóki nie zostaną zamknięte wszystkie uchwyty odwołujący się do tej kolejki, a wszystkie jednostki pracy, które mają wpływ na kolejkę, zostały zatwierdzone lub wycofane.

W systemie z/OS kolejka, która została logicznie usunięta, ale nie została jeszcze usunięta z systemu, uniemożliwia utworzenie nowej kolejki o tej samej nazwie, co usunięta kolejka. W tym przypadku wywołanie MQOPEN nie powiodło się z kodem przyczyny MQRC_NAME_IN_USE. Ponadto taka kolejka może być nadal wyświetlana za pomocą komend MQSC, nawet jeśli nie można uzyskać do niej dostępu przez aplikacje.

- Po usunięciu trwałej kolejki dynamicznej, jeśli uchwyt *Hobj* określony w wywołaniu MQCLOSE nie jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, zostanie wykonane sprawdzenie, czy identyfikator użytkownika, który był używany do sprawdzania poprawności wywołania MQOPEN, jest uprawniony do usunięcia kolejki. Jeśli w wywołaniu MQOPEN została określona opcja MQOO_ALTERNATE_USER_AUTHORITY, to sprawdzany identyfikator użytkownika to *AlternateUserId*.

To sprawdzenie nie jest wykonywane, jeśli:

- Podany uchwyt jest zwracany przez wywołanie MQOPEN, które utworzyło kolejkę.
- Usuwana kolejka jest tymczasową kolejką dynamiczną.
- Jeśli tymczasowa kolejka dynamiczna jest zamknięta, to jeśli uchwyt *Hobj* określony w wywołaniu MQCLOSE jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, kolejka jest usuwana. Taka sytuacja występuje niezależnie od opcji zamknięcia określonych w wywołaniu MQCLOSE. Jeśli w kolejce znajdują się komunikaty, są one usuwane. Nie są generowane żadne komunikaty raportów.

Jeśli istnieją niezatwierdzone jednostki pracy, które mają wpływ na kolejkę, kolejka i jej komunikaty są nadal usuwane, ale jednostki pracy nie powiodą się. Jednak, jak opisano wcześniej, zasoby powiązane z jednostkami pracy nie są zwalniane do czasu, aż każda z jednostek pracy nie zostanie zatwierdzona lub wycofana.

4. Jeśli zamykany obiekt jest *listą dystrybucyjną*, mają zastosowanie następujące punkty:

- Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest MQCO_NONE; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTIONS_ERROR lub MQRC_OPTION_NOT_VALID_FOR_TYPE, jeśli zostały określone inne opcje.
- Po zamknięciu listy dystrybucyjnej dla kolejek na liście nie są zwracane pojedyncze kody zakończenia i kody przyczyny; tylko parametry *CompCode* i *Reason* wywołania są dostępne dla celów diagnostycznych.

Jeśli wystąpi awaria podczas zamykania jednej z kolejek, menedżer kolejek kontynuuje przetwarzanie i podejmuje próbę zamknięcia pozostałych kolejek na liście dystrybucyjnej. Parametry *CompCode* i *Reason* wywołania są ustawione w taki sposób, aby zwracane były informacje opisujące błąd. Kod zakończenia jest możliwy do wywołania MQCC_FAILED, mimo że większość kolejek została pomyślnie zamknięta. Kolejka, w której wystąpił błąd, nie została zidentyfikowana.

Jeśli wystąpi awaria w więcej niż jednej kolejce, nie jest zdefiniowana, która awaria jest raportowana w parametrach *CompCode* i *Reason*.

5. W systemie IBM i, jeśli aplikacja była połączona niejawnie po wydaniu pierwszego wywołania MQOPEN, podczas wykonywania ostatniej operacji MQCLOSE występuje niejawna zmaterializowana tabela MQDISC.

Tylko aplikacje działające w trybie zgodności mogą być połączone niejawnie; inne aplikacje muszą wywołać wywołanie MQCONN lub MQCONNX w celu jawnego nawiązania połączenia z menedżerem kolejek.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Wywołanie C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQHOBJ    Hobj;      /* Object handle */
MQLONG    Options;   /* Options that control the action of MQCLOSE */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object handle
01 HOBJ       PIC S9(9) BINARY.
** Options that control the action of MQCLOSE
01 OPTIONS    PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl Options    fixed bin(31); /* Options that control the action of
                                MQCLOSE */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQCLOSE, (HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN      DS  F  Connection handle
HOBJ       DS  F  Object handle
OPTIONS    DS  F  Options that control the action of MQCLOSE
COMPCODE   DS  F  Completion code
REASON     DS  F  Reason code qualifying COMPCODE
```

Wywołanie języka Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn As Long 'Connection handle'  
Dim Hobj As Long 'Object handle'  
Dim Options As Long 'Options that control the action of MQCLOSE'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCMIT-zatwierdzanie zmian

Wywołanie MQCMIT wskazuje menedżerowi kolejek, że aplikacja osiągnęła punkt synchronizacji, oraz że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają zostać wykonane na stałe.

Komunikaty umieszczone jako część jednostki pracy są udostępniane innym aplikacjom; komunikaty pobierane jako część jednostki pracy są usuwane.

- **z/OS** W systemie z/OS wywołanie jest używane tylko przez programy wsadowe (w tym IMS wsadowe programy DL/I).
- **IBM i** W systemie IBM i wywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Ważne: **IBM i** Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Składnia

MQCMIT (*Hconn*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Podane kody przyczyn to te, które menedżer kolejek może zwrócić dla parametru *Reason*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') Wytworzona jednostka pracy.

MQRC_OUTCOME_PENDING

(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

WYWOŁANIE mqrc_call_przerwane

(2549, X'9F5') Komenda MQPUT lub MQCMIT została przerwana i przetwarzanie ponownego połączenia nie może ponownie nawiązać określonego wyniku.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

Błąd MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

MQRC_OUTCOME_MIXED

(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RECONNECT_NIE POWIODŁO SIĘ

(2548, X'9F4') Po ponownym połączeniu wystąpił błąd podczas ponownego podłączenia uchwytów dla połączenia z możliwością ponownego połączenia.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Tego wywołania należy używać tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Może to być:
 - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
 - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja [“MQBEGIN-Rozpoczęcie jednostki pracy”](#) na stronie 637.
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, zamiast komendy MQCMIT należy użyć odpowiedniego wywołania zatwierdzenia. Środowisko może również obsługiwać niejawnie zatwierdzenie spowodowane przez aplikację kończąca się normalnie.
 - W systemie z/OS należy użyć następujących wywołań:
 - Programy wsadowe (w tym wsadowe programy DL/I produktu IMS) mogą używać wywołania MQCMIT, jeśli jednostka pracy ma wpływ tylko na zasoby produktu IBM MQ . Jeśli jednak jednostka pracy ma wpływ na zasoby zarówno IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRCMIT udostępnionego przez usługę RRS (Recoverable Resource Service) produktu z/OS . Wywołanie SRRCMIT zatwierdza zmiany zasobów należących do menedżerów zasobów, które zostały włączone dla koordynacji RRS.
 - Aplikacje produktu CICS muszą używać komendy EXEC CICS SYNCPOINT , aby jawnie zatwierdzić jednostkę pracy. Alternatywnie, zakończenie transakcji spowoduje niejawnie zatwierdzenie jednostki pracy. Wywołanie MQCMIT nie może być używane w aplikacjach CICS .
 - Aplikacje produktu IMS (inne niż wsadowe programy DL/I) muszą używać wywołań programu IMS , takich jak GU i CHKP , aby zatwierdzić jednostkę pracy. Wywołanie MQCMIT nie może być używane w aplikacjach IMS (innych niż wsadowe programy DL/I).
 - W systemie IBM należy użyć tego wywołania dla lokalnych jednostek pracy koordynowanych przez menedżera kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) nie może zostać wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej informacji na ten temat zawiera sekcja [Uwagi dotyczące użycia MQDISC](#) .
4. Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwytami kolejki i obejmują takie elementy jak:
 - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset* i *MsgFlags* w strukturze MQMD.
 - Określa, czy komunikat jest częścią jednostki pracy.
 - W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Gdy jednostka pracy jest zatwierdzana, menedżer kolejek zachowuje informacje o grupie i segmencie, a aplikacja może kontynuować wprowadzanie lub pobieranie komunikatów w bieżącej grupie komunikatów lub w komunikacie logicznym.

Zachowywanie informacji o grupach i segmentach, gdy zatwierdzana jest jednostka pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy. Korzystanie z kilku jednostek pracy jest korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu. Szczegółowe informacje na temat sposobu restartowania w poprawnym punkcie po awarii systemu znajdują się w sekcji [MQPMO_LOGICAL_ORDER](#) i [MQGMO_LOGICAL_ORDER](#).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

5. Jednostka pracy ma ten sam zasięg, co uchwyt połączenia; wszystkie wywołania produktu IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Więcej informacji na temat zasięgu uchwytów połączeń zawiera opis parametru *Hconn* opisanego w tabeli MQCONN.
6. To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.
7. Długotrwa aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub żądania z powrotem, może zapętnić kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zapobiec temu, administrator musi ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapętnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duże, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.
8. W systemach UNIX i Windows , jeśli parametr *Reason* ma wartość MQRC_CONNECTION_BROKEN (z parametrem *CompCode* o wartości MQCC_FAILED) lub parametr MQRC_UNEXPECTED_ERROR jest możliwy, że jednostka pracy została pomyślnie zatwierdzona.

Wywołanie C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**      Connection handle
01 HCONN      PIC S9(9) BINARY.
**      Completion code
01 COMPCODE   PIC S9(9) BINARY.
**      Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQCMIT, (HCONN, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn As Long 'Connection handle'  
Dim CompCode As Long 'Completion code'  
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCONN-Połączenie menedżera kolejek

Wywołanie MQCONN łączy program aplikacji z menedżerem kolejek.

Udostępnia uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach kolejowania komunikatów.

- W systemie z/OS aplikacje produktu CICS nie muszą wywoływać tego wywołania. Te aplikacje są automatycznie połączone z menedżerem kolejek, z którym połączony jest system CICS. Jednak wywołania MQCONN i MQDISC są nadal akceptowane z aplikacji CICS.
- W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Te aplikacje są automatycznie połączone z menedżerem kolejek po wydaniu pierwszego wywołania MQOPEN. Jednak wywołania MQCONN i MQDISC są nadal akceptowane z aplikacji IBM i.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Inne aplikacje (czyli aplikacje, które nie działają w trybie zgodności) muszą używać wywołania MQCONN lub MQCONNX do nawiązywania połączenia z menedżerem kolejek, a wywołanie MQDISC w celu rozłączenia się z menedżerem kolejek. Jest to zalecany styl programowania.

Nie można nawiązać połączenia z klientem tylko na serwerze, a połączenie lokalne nie może zostać nawiązane w przypadku instalacji tylko klienta.

Składnia

MQCONN (*QMGrName*, *Hconn*, *CompCode*, *Przyczyna*)

Parametry

QMGrName

Typ: MQCHAR48 -dane wejściowe

Jest to nazwa menedżera kolejek, z którym aplikacja chce się połączyć. Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (od A do Z)

- Małe litery alfabetu (od a do z)
- Cyfry cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Znak o kodzie zero może być używany do wskazywania końca istotnych danych w nazwie; wartości null i dowolnych po nim znaków są traktowane jako znaki puste. W środowiskach wskazanych poniżej obowiązują następujące ograniczenia:

- W systemach, w których używane jest kodowanie EBCDIC Katakana, nie można używać małych liter.
- W systemie z/OSnazwy, których nazwy zaczynają się lub kończą się znakiem podkreślenia, nie mogą być przetwarzane przez panele kontrolne i operowe. Z tego powodu należy unikać takich nazw.
- W systemie IBM należy ująć nazwy zawierające małe litery, ukośnik lub znak procentu w cudzysłowie, jeśli zostały one określone w komendach. Nie należy określać tych znaków cudzysłowu w parametrze *QMGrName* .

Jeśli nazwa składa się całkowicie z odstępów, używana jest nazwa *domyślnego* menedżera kolejek. Należy jednak zwrócić uwagę na użycie pustych nazw menedżerów kolejek opisanych w sekcji dotyczącej aplikacji produktu IBM MQ MQI client .

Nazwa określona dla parametru *QMGrName* musi być nazwą menedżera kolejek *connectable* lub, jeśli używane są grupy menedżerów kolejek, nazwą grupy menedżerów kolejek.

W systemie z/OSmenedżery kolejek, z którymi jest możliwe nawiązanie połączenia, są określane przez środowisko:

- W przypadku produktu CICSmożna używać tylko menedżera kolejek, z którym połączony jest system CICS . Parametr *QMGrName* musi być nadal określony, ale jego wartość jest ignorowana; puste znaki są odpowiednią opcją.
- W przypadku produktu IMStylko menedżery kolejek, które są wymienione w tabeli definicji podsystemu (CSQQDEFV), i wymienione w tabeli SSM w produkcie IMS, są podłączane (patrz uwaga o składni 6).
- W przypadku zadań wsadowych z/OS i TSO tylko menedżery kolejek rezydualne w tym samym systemie, w którym znajduje się aplikacja, są podłączane (patrz uwaga dotycząca użycia 6).

Grupy współużytkowania kolejek:W systemach, w których istnieje kilka menedżerów kolejek i są one skonfigurowane w celu utworzenia grupy współużytkowania kolejki, nazwa grupy współużytkowania kolejki może zostać określona dla produktu *QMGrName* w miejsce nazwy menedżera kolejek. Umożliwia to aplikacji nawiązanie połączenia z *dowolnym* menedżerem kolejek, który jest dostępny w grupie współużytkowania kolejki i który znajduje się na tym samym obrazie produktu z/OS co aplikacja. System można również skonfigurować w taki sposób, że użycie pustego *QMGrName* łączy się z grupą współużytkowania kolejki zamiast do domyślnego menedżera kolejek.

Jeśli parametr *QMGrName* określa nazwę grupy współużytkowania kolejki, ale istnieje również menedżer kolejek o tej nazwie w systemie, to połączenie jest nawiązane z tym drugim, preferowanym względem poprzedniego. Jeśli połączenie nie powiedzie się, to połączenie z jednym z menedżerów kolejek w grupie współużytkowania kolejki nie powiedzie się.

Jeśli nawiązanie połączenia powiedzie się, można użyć uchwytu zwróconego przez wywołanie MQCONN lub MQCONNX, aby uzyskać dostęp do *wszystkich* zasobów (współużytkowanych i niewspółużytkowanych) należących do menedżera kolejek, do którego nawiązała połączenie. Dostęp do tych zasobów podlega typowym kontrolom autoryzacji.

Jeśli aplikacja generuje dwa wywołania MQCONN lub MQCONNX w celu nawiązania współbieżnych połączeń, a jeden lub oba wywołania określają nazwę grupy współużytkowania kolejki, to drugie wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_ALREADY_CONNECTED, gdy połączy się z tym samym menedżerem kolejek co pierwsze wywołanie.

Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS. Połączenie z grupą współużytkowania kolejki jest obsługiwane tylko w zadaniach wsadowych, wsadowych RRS, CICSi środowiskach TSO. W przypadku produktu CICSmożna używać tylko grupy współużytkowania

kolejek, z którą połączony jest system CICS . Nadal należy określić parametr **QMgrName** , ale jego wartość jest ignorowana; odpowiednie znaki są puste.



Ostrzeżenie: Program IMS nie może połączyć się z grupą współużytkownika kolejek.

Aplikacje produktu IBM MQ MQI client: w przypadku aplikacji produktu IBM MQ MQI client połączenie jest wykonywane dla każdej definicji kanału połączenia klienckiego z określoną nazwą menedżera kolejek, dopóki nie powiedzie się to połączenie. Menedżer kolejek musi jednak mieć taką samą nazwę, jak określona nazwa. Jeśli zostanie podana pusta nazwa, każdy kanał połączenia klienckiego z pustą nazwą menedżera kolejek zostanie wypróbowany do czasu, aż zakończy się pomyślnie. W takim przypadku nie będzie można sprawdzić rzeczywistej nazwy menedżera kolejek.

Aplikacje klienckie produktu IBM MQ nie są obsługiwane w produkcie z/OS, ale produkt z/OS może działać jako serwer IBM MQ , z którym mogą się łączyć aplikacje klienckie produktu IBM MQ .

Grupy menedżerów kolejek produktu IBM MQ MQI client: Jeśli określona nazwa rozpoczyna się od gwiazdki (*), menedżer kolejek, do którego nawiąże połączenie, może mieć inną nazwę niż nazwa określona przez aplikację. Podana nazwa (bez gwiazdki) definiuje *grupę* menedżerów kolejek, które kwalifikują się do połączenia. Implementacja wybiera jedną z grupy poprzez próbę wykonania każdego z nich do momentu znalezienia połączenia, do którego można nawiązać połączenie. Kolejność, w jakiej próby połączeń są podejmowane, zależy od wartości wagi kanału klienta i powinowactwa połączeń kanałów kandydujących. Jeśli żaden z menedżerów kolejek w grupie nie jest dostępny do połączenia, wywołanie nie powiedzie się. Każdy menedżer kolejek jest sprawdzany tylko raz. Jeśli dla nazwy została określona tylko gwiazdka, używana jest domyślna grupa menedżerów kolejek zdefiniowanych przez implementację.

Grupy menedżerów kolejek są obsługiwane tylko w przypadku aplikacji działających w środowisku klienta MQ. Wywołanie nie powiedzie się, jeśli aplikacja inna niż kliencka określa nazwę menedżera kolejek rozpoczynającą się od gwiazdki. Grupa jest definiowana przez udostępnienie kilku definicji kanału połączenia klienckiego z tą samą nazwą menedżera kolejek (określoną nazwą bez gwiazdki) w celu komunikowania się z każdym z menedżerów kolejek w grupie. Grupę domyślną definiuje się, podając co najmniej jedną definicję kanału połączenia klienckiego, każdy z pustą nazwą menedżera kolejek (podając wszystkie puste nazwy, dlatego ten sam efekt ma taki sam efekt, jak w przypadku podania jednej gwiazdki dla nazwy aplikacji klienckiej).

Po nawiązaniu połączenia z jednym menedżerem kolejek grupy aplikacja może w typowy sposób określić odstęp w polach nazwy menedżera kolejek w deskryptorach komunikatów i obiektów w celu oznaczenia nazwy menedżera kolejek, z którym aplikacja nawiązała połączenie (*lokalny menedżer kolejek*). Jeśli aplikacja musi znać tę nazwę, należy użyć wywołania MQINQ w celu uzyskania informacji o atrybucie menedżera kolejek produktu **QMgrName** .

Wstępne usunięcie gwiazdki z nazwą połączenia oznacza, że aplikacja nie jest zależna od połączenia z określonym menedżerem kolejek w grupie. Odpowiednie aplikacje to:

- Aplikacje, które umieszczają komunikaty, ale nie dostają komunikatów.
- Aplikacje, które umieszczają komunikaty żądań, a następnie otrzymują komunikaty odpowiedzi z kolejki *tymczasowej dynamicznej* .

Nieodpowiednie aplikacje to takie, które muszą pobrać komunikaty z określonej kolejki w określonym menedżerze kolejek. Aplikacje takie nie mogą prefikować nazwy za pomocą gwiazdki.

Jeśli zostanie podana gwiazdka, maksymalna długość pozostałej części nazwy to 47 znaków.

Długość tego parametru jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

Hconn

Typ: MQHCONN-wyjście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Określ go we wszystkich kolejnych wywołaniach kolejkowania komunikatów wywołanych przez aplikację. Traci ona ważność po wywołaniu wywołania MQDISC lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, kończy działanie.

Produkt IBM MQ udostępnia teraz bibliotekę mqm z pakietami klientów, a także pakiety serwera. Oznacza to, że po wywołaniu interfejsu MQI, który znajduje się w bibliotece mqm, sprawdzana jest poprawność typu połączenia, aby sprawdzić, czy jest to połączenie klienta lub serwera, a następnie wykonywane jest poprawne wywołanie bazowe. Z tego powodu wyjście, które jest przekazywane *Hconn*, może być teraz powiązane z biblioteką mqm, ale używane w instalacji klienta.

Zakres uchwytu: Zasięg zwróconego uchwytu zależy od wywołania używanego do łączenia się z menedżerem kolejek (MQCONN lub MQCONNX). Jeśli używane wywołanie to MQCONNX, zasięg uchwytu zależy również od opcji MQCNO_HANDLE_SHARE_* określonej w polu *Options* struktury MQCNO.

- Jeśli jest to wywołanie MQCONN lub podano opcję MQCNO_HANDLE_SHARE_NONE, zwrócony uchwyt jest *niewspółużytkowanym* uchwytym.

Zasięgiem niewspółużytkowanej obsługi jest najmniejsza jednostka przetwarzania równoległego obsługiwana przez platformę, na której działa aplikacja (szczegółowe informacje na ten temat zawiera sekcja [Tabela 103 na stronie 671](#)). Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołanie zostało wydane.

- Jeśli zostanie określona opcja MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwrócony uchwyt będzie *współużytkowanym* uchwytym.

Zasięgiem współużytkowanego uchwytu jest proces, który jest właścicielem wątku, z którego wywołanie zostało wydane; uchwyt może być używany z dowolnego wątku, który należy do tego procesu. Nie wszystkie platformy obsługują wątki.

- Jeśli wywołanie MQCONN lub MQCONNX nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wówczas wartość *Hconn* jest niezdefiniowana.

Tabela 103. Zasięg niewspółużytkowanych uchwytów na różnych platformach

Platforma	Zasięg niewspółużytkowanego uchwytu
z/OS	• CICS: zadanie CICS • IMS: zadanie, do następnego punktu synchronizacji (z wyłączeniem podzadań zadania) • Zadania wsadowe i TSO produktu z/OS : zadanie (z wyłączeniem podzadań zadania)
IBM i	Praca
Systemy UNIX	Wątek
16-bitowe aplikacje produktu Windows	Proces
32-bitowe aplikacje produktu Windows	Wątek

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności wartość zwracana jest następująca:

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

POŁĄCZONO MQRC_ALREADY_CONNECTED

(2002, X'7D2') Aplikacja jest już połączona.

MQRC_CLUSTER_EXIT_LOAD_ERROR

(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

MQRC_SSL_ALREADY_ZAINICJOWANY

(2391, X' 957 ') SSL zostało już zainicjowane.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR

(2129, X'851 ') Nie można załadować modułu połączenia adaptera.

Błąd MQRC_ADAPTER_DEFS_ERROR

(2131, X'853 ') Moduł definicji podsystemu adaptera nie jest poprawny.

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854 ') Nie można załadować modułu definicji podsystemu adaptera.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ADAPTER_STORAGE_NIEDOBÓR

(2127, X'84F') Niewystarczająca ilość pamięci dla adaptera.

MQRC_ANOTHER_Q_MGR_CONNECTED

(2103, X'837 ') Inny menedżer kolejek jest już połączony.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_INIT_ERROR

(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

MQRC_API_EXIT_TERM_ERROR

(2376, X' 948 ') zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CONN_ID_IN_USE

(2160, X'870 ') Identyfikator połączenia jest już używany.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_ERROR, BŁĄD

(2273, X'8E1') Błąd podczas przetwarzania wywołania MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') występuje w wywołaniu MQCONN lub MQCONNX, gdy menedżer kolejek nie może nawiązać połączenia żadanego typu połączenia w bieżącej instalacji. Nie można nawiązać połączenia z klientem na serwerze tylko do instalacji. Nie można nawiązać połączenia lokalnego w przypadku instalacji tylko klienta.

MQRC_CONNECTION_QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

Błąd MQRC_CRYPTO_HARDWARE_ERROR

(2382, X'94E') Błąd konfiguracji sprzętu szyfrującego.

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873 ') Koordynator odtwarzania istnieje.

Błąd MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

Dodatkowo, w wywołaniu MQCONNX, przekazując blok sterujący [“MQCSP-parametry zabezpieczeń”](#) na stronie 336 z aplikacji CICS lub IMS .

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') Wywołanie MQCONN zostało wysłane z klienta w celu połączenia się z menedżerem kolejek, ale próba przydzielenia konwersacji do systemu zdalnego nie powiodła się.

Niezgodność MQRC_INSTALLATION_MISMATCH

(2583, X'A17') Niezgodność między instalacją menedżera kolejek a wybraną biblioteką.

Błąd MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') Repozytorium kluczy nie jest poprawne.

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') Osiągnięta maksymalna liczba połączeń.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_OPEN_FAILED

(2137, X'859 ') Obiekt nie został otwarty pomyślnie.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SECURITY_ERROR,

(2063, X'80F') Wystąpił błąd zabezpieczeń.

MQRC_SSL_INITIALIZATION_ERROR,

(2393, X' 959 ') Błąd inicjowania SSL.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Menedżer kolejek, do którego nawiąże połączenie przy użyciu wywołania MQCONN, jest nazywany *lokalnym menedżerem kolejek*.
2. Kolejki, których właścicielem jest lokalny menedżer kolejek, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie komunikatów z tych kolejek.

Kolejki współużytkowane, których właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie komunikatów z tych kolejek.

Kolejki, których właścicielem jest zdalne menedżery kolejek, są wyświetlane jako kolejki zdalne. Możliwe jest umieszczanie komunikatów w tych kolejkach, ale nie pobieranie komunikatów z tych kolejek.
3. Jeśli menedżer kolejek zakończy się niepowodzeniem podczas działania aplikacji, aplikacja musi ponownie wywołać wywołanie MQCONN, aby uzyskać nowy uchwyt połączenia, który ma być używany podczas kolejnych wywołań produktu IBM MQ . Aplikacja może wywoływać wywołanie MQCONN okresowo, dopóki wywołanie nie powiedzie się.

Jeśli aplikacja nie jest pewna, czy jest połączona z menedżerem kolejek, aplikacja może bezpiecznie wydać wywołanie MQCONN w celu uzyskania uchwytu połączenia. Jeśli aplikacja jest już połączona, zwrócony uchwyt jest taki sam, jak zwrócony przez poprzednie wywołanie MQCONN, ale z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_ALREADY_CONNECTED.
4. Gdy aplikacja zakończy się za pomocą wywołań IBM MQ , aplikacja musi używać wywołania MQDISC do rozłączenia się z menedżerem kolejek.
5. Jeśli wywołanie MQCONN nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wówczas wartość Hconn jest niezdefiniowana.
6. wł.z/OS:
 - Aplikacje wsadowe, TSO i IMS muszą wywoływać wywołanie MQCONN w celu użycia innych wywołań produktu IBM MQ . Aplikacje te mogą łączyć się jednocześnie z więcej niż jednym menedżerem kolejek.

Jeśli menedżer kolejek nie powiedzie się, aplikacja musi wywołać wywołanie ponownie po zrestartowaniu menedżera kolejek w celu uzyskania nowego uchwytu połączenia.

Mimo że aplikacje produktu IMS mogą wielokrotnie wywoływać wywołania MQCONN, nawet jeśli są one już połączone, nie jest to zalecane w przypadku programów przetwarzania komunikatów online (MPPs).
 - Aplikacje produktu CICS nie muszą wywoływać wywołań MQCONN w celu użycia innych wywołań produktu IBM MQ , ale mogą to zrobić, jeśli chcą, zarówno wywołanie MQCONN, jak i wywołanie MQDISC są akceptowane. Nie jest jednak możliwe jednoczesne nawiązanie połączenia z więcej niż jednym menedżerem kolejek.

Jeśli menedżer kolejek nie powiedzie się, te aplikacje zostaną automatycznie ponownie połączone po restarcie menedżera kolejek, a więc nie trzeba wywoływać wywołań MQCONN.
7. W systemie z/OS, aby zdefiniować dostępne menedżery kolejek:
 - W przypadku aplikacji wsadowych programiści systemu mogą używać makra CSQBDEF do tworzenia modułu (CSQBDEFV), który definiuje domyślną nazwę menedżera kolejek lub nazwę grupy współużytkowania kolejki.
 - W przypadku aplikacji systemu IMS programiści systemu mogą używać makra CSQQDEFX do tworzenia modułu (CSQQDEFV), który definiuje nazwy dostępnych menedżerów kolejek i określa domyślny menedżer kolejek.

Ponadto każdy menedżer kolejek musi być zdefiniowany w regionie sterującym IMS i do każdego regionu zależnego uzyskanego z dostępem do tego menedżera kolejek. Aby to zrobić, należy utworzyć element podsystemu w IMS. Biblioteka PROCLIB i identyfikacja elementu podsystemu do odpowiednich regionów IMS. Jeśli aplikacja podejmie próbę nawiązania połączenia z menedżerem kolejek, który nie jest zdefiniowany w podzbiorze podsystemu dla jego regionu IMS, aplikacja będzie się abkońować.

 Więcej informacji na temat korzystania z tych makr można znaleźć w sekcji [Makra przeznaczone do użytku przez klienta](#).

8. W systemie IBM i aplikacje napisane dla poprzednich wersji menedżera kolejek mogą być uruchamiane bez ponownego kompilowania. Jest to nazywan *tryb zgodności*. Ten tryb działania udostępnia kompatybilne środowisko wykonawcze dla aplikacji. Składa się on z następujących elementów:

- Program usługowy AMQZSTUB znajdujący się w bibliotece QMQM.

AMQZSTUB udostępnia ten sam interfejs publiczny co poprzednie wydania i ma ten sam podpis. Użyj tego programu usługowego, aby uzyskać dostęp do interfejsu MQI za pomocą wywołań procedur skonsolidowanych.

- Program QMQM znajdujący się w bibliotece QMQM.

QMQM udostępnia sposób uzyskiwania dostępu do interfejsu MQI za pomocą dynamicznych wywołań programu.

- Programy MQCLOSE, MQCONN, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQPUT1 i MQSET znajdujące się w bibliotece QMQM.

Programy te udostępniają również sposoby uzyskiwania dostępu do interfejsu MQI za pomocą dynamicznych wywołań programów, ale z listą parametrów, która odpowiada standardowi opisom wywołań IBM MQ.

Te trzy interfejsy nie zawierają możliwości, które zostały wprowadzone w wersji IBM MQ 5.1. Na przykład wywołania MQBACK, MQCMIT i MQCONNX nie są obsługiwane. Obsługa udostępniana przez te interfejsy jest dostępna tylko dla aplikacji jednowątkowych.

Obsługa nowych wywołań IBM MQ w aplikacjach jednowątkowych, a także dla wszystkich wywołań IBM MQ w aplikacjach wielowątkowych, jest udostępniana za pośrednictwem programów serwisowych LIBMQM i LIBMQM_R.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

9. W systemie IBM i programy, które kończą się nieprawidłowo, nie są automatycznie odłączane od menedżera kolejek. Napisz aplikację, aby zezwolić na możliwość wywołania MQCONN lub MQCONNX zwracając kod zakończenia MQCC_WARNING i kod przyczyny MQRC_ALREADY_CONNECTED. Użyj uchwytu połączenia zwróconego w tej sytuacji jako normalnego.

Wywołanie C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQCHAR48  QMgrName;    /* Name of queue manager */
MQHCONN    Hconn;       /* Connection handle */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Name of queue manager
01 QMGRNAME PIC X(48).
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl QMgrName char(48); /* Name of queue manager */
dcl Hconn fixed bin(31); /* Connection handle */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQCONN, (QMGRNAME, HCONN, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

```
QMGRNAME DS CL48 Name of queue manager
HCONN DS F Connection handle
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

Wywołanie języka Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim QMgrName As String*48 'Name of queue manager'
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCONNX-Connect menedżer kolejek (rozszerzony)

Wywołanie MQCONNX łączy program aplikacji z menedżerem kolejek. Udostępnia uchwyt połączenia menedżera kolejek, który jest używany przez aplikację przy kolejnych wywołaniach programu IBM MQ.

Wywołanie MQCONNX jest podobne do wywołania MQCONN, z tą różnicą, że MQCONNX umożliwia określenie opcji sterujących sposobem, w jaki działa wywołanie.

- To wywołanie jest obsługiwane we wszystkich systemach IBM MQ i klientach IBM MQ połączonych z tymi systemami.
- W systemie IBM iwywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Nie można nawiązać połączenia z klientem tylko na serwerze, a połączenie lokalne nie może zostać nawiązane w przypadku instalacji tylko klienta.

Składnia

MQCONN (QMgrName, ConnectOpts, Hconn, CompCode, Przyczyna)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Szczegółowe informacje zawiera opis parametru *QMgrName* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek”](#) na stronie 668 .

ConnectOpts

Typ: MQCNO-input/output

Szczegółowe informacje na ten temat zawiera sekcja [“MQCNO-opcje połączenia”](#) na stronie 320.

Hconn

Typ: MQHCONN-wyjście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Określ go we wszystkich kolejnych wywołaniach kolejkowania komunikatów wywołanych przez aplikację. Traci ona ważność po wywołaniu wywołania MQDISC lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, kończy działanie.

Produkt IBM MQ udostępnia teraz bibliotekę mqm z pakietami klientów, a także pakiety serwera. Oznacza to, że po wywołaniu interfejsu MQI, który znajduje się w bibliotece mqm, sprawdzana jest poprawność typu połączenia, aby sprawdzić, czy jest to połączenie klienta lub serwera, a następnie wykonywane jest poprawne wywołanie bazowe. Z tego powodu wyjście, które jest przekazywane *Hconn* , może być teraz powiązane z biblioteką mqm, ale używane w instalacji klienta.

Zakres uchwytu: Zasięg zwróconego uchwytu zależy od wywołania używanego do łączenia się z menedżerem kolejek (MQCONN lub MQCONNX). Jeśli używane wywołanie to MQCONNX, zasięg uchwytu zależy również od opcji MQCNO_HANDLE_SHARE_ * określonej w polu *Options* struktury MQCNO.

- Jeśli jest to wywołanie MQCONN lub podano opcję MQCNO_HANDLE_SHARE_NONE, zwrócony uchwyt jest *niewspółużytkowanym* uchwytym.

Zasięgiem niewspółużytkowanej obsługi jest najmniejsza jednostka przetwarzania równoległego obsługiwana przez platformę, na której działa aplikacja (szczegółowe informacje na ten temat zawiera sekcja [Tabela 104](#) na stronie 678). Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołanie zostało wydane.

- Jeśli zostanie określona opcja MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwrócony uchwyt będzie *współużytkowanym* uchwytym.

Zasięgiem współużytkowanego uchwytu jest proces, który jest właścicielem wątku, z którego wywołanie zostało wydane; uchwyt może być używany z dowolnego wątku, który należy do tego procesu. Nie wszystkie platformy obsługują wątki.

- Jeśli wywołanie MQCONN lub MQCONNX nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wówczas wartość Hconn jest niezdefiniowana.

Tabela 104. Zasięg niewspółużytkowanych uchwytów na różnych platformach

Platforma	Zasięg niewspółużytkowanego uchwytu
z/OS	• CICS: zadanie CICS • IMS: zadanie, do następnego punktu synchronizacji (z wyłączeniem podzadań zadania) • Zadania wsadowe i TSO produktu z/OS : zadanie (z wyłączeniem podzadań zadania)
IBM i	Praca
Systemy UNIX	Wątek
16-bitowe aplikacje produktu Windows	Proces
32-bitowe aplikacje produktu Windows	Wątek

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności wartość zwracana jest następująca:

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

CompCode

Typ: MQLONG-wyjście

Szczegółowe informacje zawiera opis parametru *CompCode* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek”](#) na stronie 668 .

reason

Typ: MQLONG-wyjście

Następujące kody mogą być zwracane przez wywołania MQCONN i MQCONNX. Aby uzyskać listę dodatkowych kodów, które mogą zostać zwrócone przez wywołanie MQCONNX, należy zapoznać się z następującymi kodami.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

POŁĄCZONO MQRC_ALREADY_CONNECTED

(2002, X'7D2') Aplikacja jest już połączona.

MQRC_CLUSTER_EXIT_LOAD_ERROR

(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

MQRC_SSL_ALREADY_ZAINICJOWANY

(2391, X' 957 ') SSL zostało już zainicjowane.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR

(2129, X'851 ') Nie można załadować modułu połączenia adaptera.

Błąd MQRC_ADAPTER_DEFS_ERROR

(2131, X'853 ') Moduł definicji podsystemu adaptera nie jest poprawny.

MQRC_ADAPTER_DEFS_LOAD_ERROR

(2132, X'854 ') Nie można załadować modułu definicji podsystemu adaptera.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ADAPTER_STORAGE_NIEDOBÓR

(2127, X'84F') Niewystarczająca ilość pamięci dla adaptera.

MQRC_ANOTHER_Q_MGR_CONNECTED

(2103, X'837 ') Inny menedżer kolejek jest już połączony.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_INIT_ERROR

(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

MQRC_API_EXIT_TERM_ERROR

(2376, X' 948 ') zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CONN_ID_IN_USE

(2160, X'870 ') Identyfikator połączenia jest już używany.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_ERROR, BŁĄD

(2273, X'8E1') Błąd podczas przetwarzania wywołania MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE

(2568, X'A08') występuje w wywołaniu MQCONN lub MQCONN, gdy menedżer kolejek nie może nawiązać połączenia żadanego typu połączenia w bieżącej instalacji. Nie można nawiązać połączenia z klientem na serwerze tylko do instalacji. Nie można nawiązać połączenia lokalnego w przypadku instalacji tylko klienta.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

Błąd MQRC_CRYPTOHARDWARE_ERROR

(2382, X'94E') Błąd konfiguracji sprzętu szyfrującego.

MQRC_DUPLICATE_RECOV_COORD

(2163, X'873 ') Koordynator odtwarzania istnieje.

Błąd MQRC_ENVIRONMENT_ERROR

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

Dodatkowo, w wywołaniu MQCONN, przekazując blok sterujący [“MQCSP-parametry zabezpieczeń”](#) na stronie 336 z aplikacji CICS lub IMS .

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

MQRC_HOST_NOT_AVAILABLE

(2538, X'9EA') Wywołanie MQCONN zostało wysłane z klienta w celu połączenia się z menedżerem kolejek, ale próba przydzielenia konwersacji do systemu zdalnego nie powiodła się.

Niezgodność MQRC_INSTALLATION_MISMATCH

(2583, X'A17') Niezgodność między instalacją menedżera kolejek a wybraną biblioteką.

Błąd MQRC_KEY_REPOSITORY_ERROR

(2381, X'94D') Repozytorium kluczy nie jest poprawne.

MQRC_MAX_CONNS_LIMIT_REACHED

(2025, X'7E9') Osiągana maksymalna liczba połączeń.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_OPEN_FAILED

(2137, X'859 ') Obiekt nie został otwarty pomyślnie.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SECURITY_ERROR,

(2063, X'80F') Wystąpił błąd zabezpieczeń.

MQRC_SSL_INITIALIZATION_ERROR,

(2393, X' 959 ') Błąd inicjowania SSL.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Wywołanie MQCONNX może zwrócić następujące dodatkowe kody przyczyny:

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_AIR_ERROR,

(2385, X' 951 ') Rekord informacji uwierzytelniającej nie jest poprawny.

MQRC_AUTH_INFO_CONN_NAME_ERROR

(2387, X' 953 ') Nazwa połączenia informacji uwierzytelniającej jest niepoprawna.

MQRC_AUTH_INFO_REC_COUNT_ERROR

(2383, X'94F') Liczba rekordów informacji uwierzytelniających nie jest poprawna.

MQRC_AUTH_INFO_REC_ERROR

(2384, X' 950 ') Pola rekordu informacji uwierzytelniającej nie są poprawne.

MQRC_AUTH_INFO_TYPE_ERROR

(2386, X' 952 ') Typ informacji uwierzytelniającej nie jest poprawny.

BŁĄD MQRC_CD_ERROR

(2277, X'8E5') Definicja kanału nie jest poprawna.

BŁĄD MQRC_CLIENT_CONN_ERROR

(2278, X'8E6') Pola połączenia klienta nie są poprawne.

BŁĄD MQRC_CNO_ERROR

(2139, X'85B') Struktura opcji Connect nie jest poprawna.

MQRC_CONN_TAG_IN_USE

(2271, X'8DF') Znacznik połączenia w użyciu.

MQRC_CONN_TAG_NOT_USABLE

(2350, X'92E') Znacznik połączenia nie nadaje się do użycia.

MQRC_LDAP_PASSWORD_ERROR

(2390, X' 956 ') Hasło LDAP nie jest poprawne.

MQRC_LDAP_USER_NAME_ERROR

(2388, X' 954 ') pola nazwy użytkownika LDAP nie są poprawne.

MQRC_LDAP_USER_NAME_LENGTH_ERR

(2389, X' 955 ') długość nazwy użytkownika LDAP nie jest poprawna.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

MQRC_SCO_ERROR,

(2380, X'94C') Struktura opcji konfiguracji SSL nie jest poprawna.

BŁĄD MQRC_SSL_CONFIG_ERROR

(2392, X' 958 ') Błąd konfiguracji SSL.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

W przypadku języka programowania Visual Basic stosuje się następujący punkt:

- Parametr *ConnectOpts* jest zadeklarowany jako typ MQCNO. Jeśli aplikacja działa jako IBM MQ MQI clienta wymagane jest określenie parametrów kanału połączenia klienckiego, należy zadeklarować parametr *ConnectOpts* jako typ Any, tak aby aplikacja mogła określić strukturę MQCNOCD w wywołaniu w miejscu struktury MQCNO. Oznacza to jednak, że nie można sprawdzić parametru *ConnectOpts*, aby upewnić się, że jest to poprawny typ danych.

Wywołanie C

```
MQCONN (QMGrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie języka COBOL

```
CALL 'MQCONN' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Name of queue manager
01 QMGRNAME      PIC X(48).
** Options that control the action of 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.
```

Wywołanie PL/I

```
call MQCONN (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dc1 QMgrName      char(48);      /* Name of queue manager */
dc1 ConnectOpts   like MQCNO;    /* Options that control the action of
                                MQCONN */
dc1 Hconn         fixed bin(31);  /* Connection handle */
dc1 CompCode      fixed bin(31);  /* Completion code */
dc1 Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQCONN, (QMGRNAME, CONNECTOPTS, HCONN, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

QMGRNAME	DS	CL48	Name of queue manager
CONNECTOPTS	CMQCN0A	,	Options that control the action of MQCONN
HCONN	DS	F	Connection handle
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```
MQCONN QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim QMgrName As String*48 'Name of queue manager'
Dim ConnectOpts As MQCNO 'Options that control the action of'
                          'MQCONN'
Dim Hconn As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason As Long 'Reason code qualifying CompCode'
```

MQCRTMH-Tworzenie uchwytu komunikatu

Wywołanie MQCRTMH zwraca uchwyt komunikatu.

Aplikacja może korzystać z wywołania MQCRTMH w kolejnych wywołaniach kolejkowania komunikatów:

- Użyj wywołania [MQSETMP](#) , aby ustawić właściwość uchwytu komunikatu.
- Użyj wywołania [MQINQMP](#) , aby dowiedzieć się o wartości właściwości uchwytu komunikatu.
- Za pomocą wywołania [MQDLTMP](#) można usunąć właściwość uchwytu komunikatu.

Uchwyt komunikatu może być używany w wywołaniach MQPUT i MQPUT1 w celu powiązania właściwości uchwytu komunikatu z tymi, które są umieszczane w umieszczonym komunikacie. Podobnie, określając uchwyt komunikatu w wywołaniu MQGET, właściwości pobieranego komunikatu można uzyskać za pomocą uchwytu komunikatu po zakończeniu wywołania MQGET.

Użyj komendy [MQDLTMH](#) , aby usunąć uchwyt komunikatu.

Składnia

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. Jeśli połączenie z menedżerem kolejek przestanie być poprawne, a w uchwycie komunikatu nie działa żaden wywołanie produktu IBM MQ, komenda MQDLTMH jest niejawnie wywoływana w celu usunięcia komunikatu.

Alternatywnie można podać następującą wartość:

MQHC_UNASSOCIATED_HCONN

Uchwyt połączenia nie reprezentuje połączenia z żadnym określonym menedżerem kolejek.

Jeśli ta wartość jest używana, uchwyt komunikatu musi zostać usunięty z jawnym wywołaniem komendy MQDLTMH w celu zwolnienia przydzielonej pamięci masowej; IBM MQ nigdy niejawnie usuwa uchwyt komunikatu.

Musi istnieć co najmniej jedno poprawne połączenie z menedżerem kolejek, który został utworzony w wątku tworzący uchwyt komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd MQRC_HCONN_ERROR.

W środowisku z wieloma instalacjami w jednym systemie wartość MQHC_UNASSOCIATED_HCONN jest ograniczona do użycia z pierwszą instalacją załadowaną do tego procesu. Kod przyczyny MQRC_HMSG_NOT_AVAILABLE jest zwracany, jeśli uchwyt komunikatu jest dostarczany do innej instalacji.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a ponadto można określić następującą wartość dla *Hconn* :

MQHC_DEF_CONN

Domyślny uchwyt połączenia

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOptyCrtMsg

Typ: MQCMHO-wejście

Opcje, które sterują działaniem MQCRTMH. Szczegółowe informacje na ten temat zawiera sekcja [MQCMHO](#).

Hmsg

Typ: MQHMSG-wyjście

Na wyjściu zwracany jest uchwyt komunikatu, który może być używany do ustawiania, sprawdzania i usuwania właściwości uchwytu komunikatu. Początkowo uchwyt komunikatu nie zawiera żadnych właściwości.

Uchwyt komunikatu ma również powiązany deskryptor komunikatu. Początkowo zawiera ona wartości domyślne. Wartości powiązanych pól deskryptora komunikatu można ustawiać i pytać przy użyciu wywołań MQSETMP i MQINQMP. Wywołanie MQDLTMP resetuje pole deskryptora komunikatu z powrotem do jego wartości domyślnej.

Jeśli parametr *Hconn* jest określony jako wartość MQHC_UNASSOCIATED_HCONN, to zwrócony uchwyt komunikatu może być używany w wywołaniach MQGET, MQPUT lub MQPUT1 z dowolnym połączeniem w jednostce przetwarzania, ale może być używany tylko przez jedno wywołanie produktu IBM MQ naraz. Jeśli uchwyt jest używany, gdy druga próba wywołania IBM MQ próbuje użyć tego samego uchwytu komunikatu, drugie wywołanie IBM MQ nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_MSG_HANDLE_IN_USE.

Jeśli parametr *Hconn* nie jest parametrem MQHC_UNASSOCIATED_HCONN, to zwrócony uchwyt komunikatu może być używany tylko dla określonego połączenia.

Ta sama wartość parametru *Hconn* musi być używana w kolejnych wywołaniach MQI, w których ten uchwyt komunikatu jest używany:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

Zwrócony uchwyt komunikatu przestaje być poprawny, gdy dla uchwytu komunikatu zostanie wywołane wywołanie MQDLTMH lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostanie zakończona. Komenda MQDLTMH jest wywoływana niejawnie, jeśli podczas tworzenia uchwytu komunikatu określone jest konkretne połączenie, a połączenie z menedżerem kolejek przestaje być poprawne, na przykład jeśli wywołano komendę MQDBC.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_CMHO_ERROR

(2461, X'099D') Struktura opcji uchwytu komunikatu nie jest poprawna.

MQRC_CONNECTION_BROKEN

(2273, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'07E1') Nie ma więcej dostępnych uchwytów.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;          /* Connection handle */
MQCMHO   CrtMsgHOpts;    /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;           /* Message handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

COBOL

```
CALL 'MQCRTMH' USING HCONN, CRTMSGOPTS, HMSG, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** 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);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CrtMsgHOpts like MQCMHO; /* Options that control the action of MQCRTMH */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

High Level Assembler

```
CALL MQCRTMH, (HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN      DS      F      Connection handle
CRTMSGHOPTS CMQCMHOA , Options that control the action of MQCRTMH
```

HMSG	DS	D	Message handle
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCTL-wywołania zwrotne sterowania

Wywołanie MQCTL wykonuje operacje sterujące dla wywołań zwrotnych i uchwytów obiektów otwartych dla połączenia.

Składnia

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a ponadto można określić następującą wartość specjalną dla produktu *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

// operacji

Typ: MQLONG-wejście

Operacja jest przetwarzana dla wywołania zwrotnego zdefiniowanego dla podanego uchwytu obiektu. Należy określić jedną i tylko jedną z następujących opcji:

MQOP_START,

Uruchamia konsumowanie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Wywołania zwrotne są uruchamiane w wątku uruchomionym przez system, który różni się od żadnego z wątków aplikacji.

Ta operacja umożliwia sterowanie udostępnionym uchwytem połączenia z systemem. Jedynymi wywołaniami MQI, które mogą być wysyłane przez wątek inny niż wątek konsumenta, są:

- MQCTL z operacją MQOP_STOP
- MQCTL z operacją MQOP_SUSPEND
- MQDISC-Performs MQCTL z operacją MQOP_STOP przed rozłączonym połączeniem z HConn.

Wartość MQRC_HCONN_ASYNC_ACTIVE jest zwracana, jeśli wywołanie funkcji API produktu IBM MQ zostało wysłane podczas uruchamiania uchwytu połączenia, a wywołanie nie pochodzi z funkcji konsumenta komunikatów.

Jeśli konsument komunikatów zatrzyma połączenie podczas wywołania MQCBCT_START_CALL, wywołanie MQCTL zwraca kod przyczyny niepowodzenia MQRC_CONNECTION_STOPPED.

Może to być wydane w funkcji konsumenta. W przypadku tego samego połączenia, co procedura zwrotna, jej jedynym celem jest anulowanie poprzednio wywołanej operacji MQOP_STOP.

Ta opcja nie jest obsługiwana w następujących środowiskach: CICS w systemie z/OS lub jeśli aplikacja jest powiązana z niewielowątkową biblioteką IBM MQ .

MQOP_START_WAIT

Uruchamia konsumowanie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Odbiorcy komunikatów działają w tym samym wątku, a sterowanie nie jest zwracane do programu wywołującego MQCTL, dopóki:

- Zwolniony przez użycie operacji MQCTL MQOP_STOP lub MQOP_SUSPEND, lub
- Wszystkie procedury konsumenckie zostały wyrejestrowywane lub zawieszone.

Jeśli wszyscy konsumenci zostaną wyrejestrowani lub zawieszeni, zostanie wykonana niejawną operacja MQOP_STOP.

Ta opcja nie może być używana w ramach procedury zwrotnej ani dla bieżącego uchwytu połączenia, ani dla żadnego innego uchwytu połączenia. Jeśli wywołanie jest wykonywane, zwraca wartość MQRC_ENVIRONMENT_ERROR.

Jeśli w dowolnym momencie podczas operacji MQOP_START_WAIT nie ma zarejestrowanych, niezawieszonych konsumentów wywołanie nie powiedzie się i zostanie uruchomiony kod przyczyny MQRC_NO_CALLBACKS_ACTIVE.

Jeśli podczas operacji MQOP_START_WAIT połączenie zostanie zawieszone, wywołanie MQCTL zwróci kod przyczyny ostrzeżenia o wartości MQRC_CONNECTION_SUSPENDED; połączenie pozostanie uruchomione.

Aplikacja może wydać komendę MQOP_STOP lub MQOP_RESUME. W tej instancji bloki operacji MQOP_RESUME.

Ta opcja nie jest obsługiwana w przypadku pojedynczego klienta wielowątkowego.

MQOP_STOP

Przed zakończeniem tej opcji zatrzymaj korzystanie z komunikatów i poczekaj na zakończenie operacji przez wszystkich konsumentów. Ta operacja zwalnia uchwyt połączenia.

Jeśli ta opcja jest wydawana w ramach procedury zwrotnej, ta opcja nie jest uwzględniana do czasu zakończenia procedury. Po zakończeniu procedur konsumenckich dla komunikatów, które zostały już odczytane, nie są wywoływane żadne procedury dla konsumenta komunikatów. Po zatrzymaniu wywołań zwrotnych (jeśli jest to wymagane) do procedur zwrotnych, zostaną wykonane procedury.

Jeśli jest ona wystawiona poza procedurą wywołania zwrotnego, sterowanie nie jest zwracane do programu wywołującego, dopóki nie zostaną wykonane procedury konsumenckie dla komunikatów, które zostały już odczytane, i po zatrzymaniu wywołań zwrotnych (jeśli jest wymagane) do wywołań zwrotnych. Jednak same procedury zwrotne pozostają zarejestrowane.

Ta funkcja nie ma wpływu na komunikaty odczytu z wyprzedzeniem. Należy upewnić się, że konsumenci uruchamiają komendę MQCLOSE (MQCO_QUIESCE), z poziomu funkcji zwrotnej, aby określić, czy istnieją dalsze komunikaty, które można dostarczyć.

MQOP_SUSPEND

Wstrzymaj korzystanie z komunikatów. Ta operacja zwalnia uchwyt połączenia.

Nie ma to żadnego wpływu na odczyt z wyprzedzeniem komunikatów dla aplikacji. Jeśli użytkownik zamierza zaprzestać używania komunikatów przez długi czas, należy rozważyć zamknięcie kolejki i ponowne jej otwarcie, gdy zużycie będzie kontynuowane.

Jeśli jest ona wydana z poziomu procedury zwrotnej, nie jest ona skuteczna, dopóki procedura nie zostanie zakończona. Po zakończeniu bieżących procedur zewnętrznych nie będą wywoływane żadne procedury konsumenta komunikatów.

Jeśli zostanie ona wydana poza wywołaniem zwrotnym, sterowanie nie zostanie zwrócone do programu wywołującego, dopóki nie zostanie zakończona bieżąca procedura konsumenta i nie zostanie już wywołana żadna wartość.

MQOP_RESUME

Wznów korzystanie z komunikatów.

Ta opcja jest zwykle wydawana z głównego wątku aplikacji, ale może być również używana w ramach procedury zwrotnej w celu anulowania wcześniejszego żądania zawieszenia wydanego w tej samej procedurze.

Jeśli wartość MQOP_RESUME jest używana do wznowienia operacji MQOP_START_WAIT, to bloki operacji są wznowiane.

ControlOpts

Typ: MQCTLO-dane wejściowe

Opcje sterujące działaniem komendy MQCTL

Szczegółowe informacje na temat struktury można znaleźć w sekcji [MQCTLO](#) .

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

Błąd MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALLBACK_LINK_ERROR

(2487, X'9B7') Nie można wywołać procedury zwrotnej

ZAREJESTROWANO MQRC_CALLBACK_NOT_

(2448, X' 990 ') Nie można wyrejestrować, zawiesić ani wznowić, ponieważ nie ma zarejestrowanej procedury zwrotnej.

MQRC_CALLBACK_ROUTINE_ERROR

(2486, X'9B6') Albo zarówno CallbackFunction , jak i CallbackName , zostały określone w wywołaniu MQOP_REGISTER.

Albo parametr CallbackFunction lub CallbackName został określony, ale nie jest zgodny z aktualnie zarejestrowaną funkcją zwrotną.

MQRC_CALLBACK_TYPE_ERROR

(2483, X'9B3') Niepoprawne pole typu CallBackType.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CBD_ERROR

(2444, X'98C') Blok opcji jest niepoprawny.

MQRC_CBD_OPTIONS_ERROR

(2484, X'9B4') Niepoprawne pole opcji MQCBD.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_CORREL_ID_ERROR (BŁĄD)

(2207, X'89F') Błąd korelacji identyfikatora.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_GET_INHIBITED

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

BŁĄD MQRC_GMO_ERROR

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

MQRC_HANDLE_IN_USE_DLA_UOW

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

MQRC_INCONSISTENT_BROWSE

(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

BŁĄD MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Opcje zgodności nie są poprawne.

MQRC_MAX_MSG_LENGTH_ERROR

(2485, X'9B5') Niepoprawne pole długości MaxMsg

Błąd MQRC_MD_ERROR

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

MQRC_MODULE_ENTRY_NOT_FOUND

(2497, X'9C1') Określony punkt wejścia funkcji nie został znaleziony w module.

Niepoprawna wartość MQRC_MODULE_INVALID

(2496, X'9C0') Moduł został znaleziony, ale ma niepoprawny typ (32 bit/64 bit) lub nie jest poprawną biblioteką dll.

MQRC_MODULE_NOT_FOUND

(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie został autoryzowany do załadowania.

BŁĄD MQRC_MSG_ID_ERROR

(2206, X'89E') Błąd identyfikatora komunikatu.

MQRC_MSG_SEQ_NUMBER_ERROR,

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') Użycie znacznika komunikatu nie jest poprawne.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

BŁĄD MQRC_OPERATION_ERROR

(2488, X'9B8') Niepoprawny kod operacji w wywołaniu API Call

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815 ') Signal outstanding for this handle.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Procedury wywołania zwrotnego muszą sprawdzać odpowiedzi ze wszystkich wywołanych przez nie usług, a jeśli procedura wykryje warunek, którego nie można rozwiązać, musi wydać komendę MQCB MQOP_DEREGISTER, aby zapobiec powtórzonemu wywołaniu procedury zwrotnej.
2. W przypadku korzystania z asynchronicznego wykorzystania w aplikacji, w której menedżer transakcji XA zarządza transakcjami globalnymi, w tym aktualizacjami do produktu IBM MQ, należy rozważyć następujące dodatkowe punkty:
 - a. Nie można wywołać komendy MQCTL (MQOP_START) dla partycji **HConn**, po jej utworzeniu, po wywołaniu programu **xa_open**.

Jest to spowodowane tym, że produkt **HConn** stał się przyłączony do kontekstu XA i dlatego nie można uzyskać do niego dostępu w oddzielnym wątku lub wątkach, które są używane przez mechanizm asynchronicznego wykorzystania.
 - b. W przypadku wywołania komendy MQCTL (MQOP_START) w tym scenariuszu wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ASYNC_XA_CONFLICT (2350).
 - c. Po wywołaniu programu **xa_openn** należy wywołać komendę MQCTL (MQOP_START_WAIT) dla partycji **HConn** po jej utworzeniu.

Jest to spowodowane tym, że ta metoda uruchamiania mechanizmu asynchronicznego konsumowania powoduje, że wszystkie kolejne wywołania zwrotne dla **HConn** są uruchamiane w wątku, w którym jest nawiązywać wywołanie MQCTL. Dlatego łącze między **HConn** i wątkiem nie jest tracone.
3.  W systemie z/OS, gdy operacja jest operacją MQOP_START:
 - Programs which use asynchronous callback routines must be authorized to use z/OS UNIX System Services (USS).
 - Programy środowiska językowego (LE), które korzystają z asynchronicznych procedur zwrotnych, muszą korzystać z opcji środowiska wykonawczego LE POSIX(ON).
 - Programy inne niż LE, które korzystają z asynchronicznych procedur zwrotnych, nie mogą korzystać z interfejsu USS pthread_create (usługa wywoływalna BPX1PTC).
4.  Obiekt MQCTL nie jest obsługiwany w ramach adaptera IMS.

Uwaga: W produkcie CICS komenda MQOP_START nie jest obsługiwana. Zamiast tego należy użyć wywołania funkcji MQOP_START_WAIT.

Wywołanie C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */
MQLONG   Operation;       /* Operation being processed */
MQCTLO   ControlOpts     /* Options that control the action of MQCTL */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Operation
01 OPERATION PIC S9(9) BINARY.
** Control Options
01 CTLOPTS.
   COPY CMQCTLOV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Operation  fixed bin(31); /* Operation */
dcl CtlOpts    like MQCTLO;   /* Options that control the action of MQCTL */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

MQDISC-rozłączenie menedżera kolejek

Wywołanie MQDISC przerywa połączenie między menedżerem kolejek a programem użytkowym. Jest to odwrót wywołania MQCONN lub MQCONNX.

- W systemie z/OSwszystkie aplikacje, które korzystają z asynchronicznego wykorzystania komunikatów, obsługi zdarzeń lub wywołań zwrotnych, główny wątek sterujący musi wywołać wywołanie MQDISC przed zakończeniem. Więcej informacji na ten temat zawiera sekcja [Asynchroniczne wykorzystanie komunikatów produktu IBM MQ](#).
- W systemie z/OSaplikacje produktu CICS nie muszą wywoływać tego wywołania w celu rozłączenia się z menedżerem kolejek.

Jeśli aplikacja CICS wykonuje to wywołanie, nie ma ona żadnego wpływu, chyba że wykonano wcześniejsze wywołanie MQCONNX, określając jedną z następujących wartości:

```
MQCNO_SERIALIZE_CONN_TAG_Q_MGR
MQCNO_SERIALIZE_CONN_TAG_QSG
```


MQCNO_RESTRICT_CONN_TAG_Q_MGR lub
MQCNO_RESTRICT_CONN_TAG_QSG

W takim przypadku wszystkie aktualnie otwarte uchwytów są zamykane.

- W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Więcej informacji zawiera temat [“MQCONN-Połączenie menedżera kolejek”](#) na stronie 668.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilewanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Składnia

MQDISC (*Hconn*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście/wyjście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN i podać następującą wartość dla produktu *Hconn*:

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia wartość *Hconn* na wartość, która nie jest poprawnym uchwytem dla środowiska. Ta wartość jest następująca:

MQHC_UNUSABLE_HCONN

Uchwyt połączenia bez użycia.

W systemie z/OS wartość *Hconn* jest ustawiana na wartość, która jest niezdefiniowana.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_BACKED_OUT

(2003, X'7D3') Wytworzona jednostka pracy.

MQRC_CONN_TAG_NOT_ZWOLNIONY

(2344, X' 928 ') Znacznik połączenia nie został zwolniony.

MQRC_OUTCOME_PENDING

(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_DISC_LOAD_ERROR

(2138, X'85A') Nie można załadować modułu rozłączenia adaptera.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_INIT_ERROR

(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

MQRC_API_EXIT_TERM_ERROR

(2376, X' 948 ') zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

MQRC_OUTCOME_MIXED

(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Jeśli wywołanie MQDISC jest wysyłane, gdy połączenie nadal ma obiekty otwarte w ramach tego połączenia, menedżer kolejek zamyka te obiekty, a opcje zamknięcia są ustawione na wartość MQCO_NONE.

2. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, dyspozycja tych zmian zależy od tego, w jaki sposób aplikacja kończy pracę:
- Jeśli aplikacja wysyła wywołanie MQDISC przed zakończeniem działania:
 - W przypadku jednostki pracy koordynowanej przez menedżer kolejek, menedżer kolejek wysyła wywołanie MQCMIT w imieniu aplikacji. Jeśli jest to możliwe, jednostka pracy jest zatwierdzana i wycofana, jeśli nie jest dostępna.
 - W przypadku zewnętrznie koordynowanej jednostki pracy nie ma zmian w statusie jednostki pracy. Jednak menedżer kolejek zwykle wskazuje, że jednostka pracy musi zostać zatwierdzona przez koordynatora jednostki pracy.

W systemach z/OS, CICS, IMS (inne niż wsadowe programy DL/1) i aplikacje RRS są podobne do tej.
 - Jeśli aplikacja kończy się normalnie, ale bez wywoływania wywołania MQDISC, działanie jest zależne od środowiska:
 - W systemie z/OS, z wyjątkiem aplikacji produktu MQ Java lub MQ JMS , wykonywane są działania opisane w uwadze 2a .
 - We wszystkich innych przypadkach czynności opisane w uwadze 2c występują.

Ze względu na różnice między środowiskami należy upewnić się, że aplikacje, które mają być portowane, albo zatwierdzają, albo wycofują jednostkę pracy przed ich zakończeniem.
 - Jeśli aplikacja zakończy działanie *nieprawidłowo* bez wywoływania wywołania MQDISC, wycofana jest jednostka pracy.
3. W systemie z/OS mają zastosowanie następujące punkty:
- Aplikacje produktu CICS nie muszą wywoływać wywołania MQDISC w celu rozłączenia się z menedżerem kolejek, ponieważ sam system CICS łączy się z menedżerem kolejek, a wywołanie MQDISC nie ma wpływu na to połączenie.
 - CICS, IMS (inne niż wsadowe programy DL/1), a aplikacje RRS używają jednostek pracy, które są koordynowane przez zewnętrznego koordynatora jednostek pracy. W wyniku tego wywołanie MQDISC nie ma wpływu na status jednostki pracy (jeśli istnieje), która istnieje w momencie wywołania wywołania.
- Jednak wywołanie MQDISC *nie* oznacza koniec używania znacznika połączenia *ConnTag* , który był powiązany z połączeniem przez wcześniejsze wywołanie MQCONN, które zostało wydane przez aplikację. Jeśli istnieje aktywna jednostka pracy, która odwołuje się do znacznika połączenia po wywołaniu wywołania MQDISC, wywołanie kończy się kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_CONN_TAG_NOT_ZWOLNIONA. Znacznik połączenia nie staje się dostępny do ponownego wykorzystania, dopóki zewnętrzny koordynator jednostki pracy nie rozwiąże jednostki pracy.
4. W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Więcej informacji można znaleźć w wywołaniu MQCONN.
- Uwaga:** W produkcie CICS komenda MQOP_START nie jest obsługiwana. Zamiast tego należy użyć wywołania funkcji MQOP_START_WAIT.

Wywołanie C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQLONG   CompCode;   /* Completion code */
MQLONG   Reason;     /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie assemblera System/390

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN      DS F Connection handle
COMPCODE   DS F Completion code
REASON     DS F Reason code qualifying COMPCODE
```

Wywołanie języka Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn      As Long 'Connection handle'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'
```

MQDLTMH-usuwanie uchwytu komunikatu

Wywołanie MQDLTMH usuwa uchwyt komunikatu i jest odwrotnym wywołaniem wywołania MQCRTMH.

Składnia

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytym połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy MQHC_UNASSOCIATED_HCONN, konieczne jest nawiązanie poprawnego połączenia w wątku usuwania uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się i nie zostanie nawiązane połączenie z opcją MQRC_CONNECTION_BROKEN.

Hmsg

Typ: MQHMSG-input/output

To jest uchwyt komunikatu, który ma zostać usunięty. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

Po pomyślnym zakończeniu wywołania uchwyt jest ustawiany na niepoprawną wartość dla środowiska. Ta wartość jest następująca:

MQHM_UNUSABLE_HMSG

Uchwyt komunikatu nie do użycia.

Uchwyt komunikatu nie może zostać usunięty, jeśli w toku jest inne wywołanie produktu IBM MQ, które zostało przekazane temu samemu uchwycie komunikatu.

HOptyDltMsg

Typ: MQDMHO-wejście

Szczegółowe informacje na ten temat zawiera sekcja [MQDMHO](#).

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DMHO_ERROR

(2462, X'099E') Struktura opcji uchwytu komunikatu usuwania nie jest poprawna.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgH0pts, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;          /* Connection handle */
MQHMSG    Hmsg;          /* Message handle */
MQDMHO    DltMsgH0pts;   /* Options that control the action of MQDLTMH */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01  HCONN    PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01  DLTMSGHOPTS.
COPY CMQDLMH0V.

** Completion code
01  COMPCODE  PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01  REASON    PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgH0pts, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn          /* Connection handle */
dcl Hmsg           /* Message handle */
dcl DltMsgH0pts like MQDMHO; /* Options that control the action of MQDLTMH */
dcl CompCode       /* Completion code */
dcl Reason         /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQDLTMH, (HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTMSGHOPTS	CMQDMHOA	,	Options that control the action of MQDLTMH
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQDLTMP-usunięcie właściwości komunikatu

Wywołanie MQDLTMP usuwa właściwość z uchwytu komunikatu i jest odwrotną wartością wywołania MQSETMP.

Składnia

MQDLTMP (*Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość musi być zgodna z uchwytym połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy MQHC_UNASSOCIATED_HCONN, należy ustawić poprawne połączenie w wątku usuwania uchwytu komunikatu.

W przeciwnym razie wywołanie nie powiedzie się i zostanie zerwane połączenie z opcją MQRC_CONNECTION_BROKEN.

Hmsg

Typ: MQHMSG-wejście

Jest to uchwyt komunikatu zawierający właściwość do usunięcia. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

DltPropOpts

Typ: MQDMPO-wejście

Szczegółowe informacje zawiera opis typu danych [MQDMPO](#).

nazwa

Typ: MQCHARV-wejście

Nazwa właściwości do usunięcia. Więcej informacji na temat nazw właściwości zawiera sekcja [Nazwy właściwości](#).

Znaki wieloznaczne nie są dozwolone w nazwie właściwości.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') Właściwość nie jest dostępna.

Błąd formatu MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'086D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DMPO_ERROR

(2481, X'09B1') Struktura opcji usuwania właściwości komunikatu nie jest poprawna.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

Błąd MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Niepoprawna nazwa właściwości.

MQRC_SOURCE_CCSID_ERROR, BŁĄD

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w:

- [Komunikaty i kody przyczyny dla produktu IBM MQ for z/OS](#)
- [Kody zakończenia i przyczyny interfejsu API dla innych platform IBM MQ](#)

.

Wywołanie C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:


```

MQHCONN Hconn;      /* Connection handle */
MQHMSG  Hmsg;       /* Message handle */
MQDMPO  DltPropOpts; /* Options that control the action of MQDLTMP */
MQCHARV Name;       /* Property name */
MQLONG  CompCode;   /* Completion code */
MQLONG  Reason;     /* Reason code qualifying CompCode */

```

Wywołanie języka COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```

** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Message handle
01 HMSG     PIC S9(18) BINARY.
** Options that control the action of MQDLTMP
01 DLTPROPOPTS.
   COPY CMQDMPOV.
** Property name
01 NAME     PIC S9(18) BINARY.
   COPY CMQCHARVV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.

```

Wywołanie PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl DltPropOpts like MQDMPO;  /* Options that control the action of MQDLTMP */
dcl Name       like MQCHARV;  /* Property name */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

Wywołanie High Level Assembler

```
CALL MQDLTMP, (HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
DLTPROPOPTS	CMQDMPOA	,	Options that control the action of MQDLTMP
NAME	CMQCHRVA	,	Property name
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQGET-Pobieranie komunikatu

Wywołanie MQGET pobiera komunikat z kolejki lokalnej, który został otwarty przy użyciu wywołania MQOPEN.

Składnia

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Hobj

Typ: MQHOBJ-wejście

Ten uchwyt reprezentuje kolejkę, z której ma zostać pobrany komunikat. Wartość *Hobj* została zwrócona przez poprzednie wywołanie MQOPEN. Kolejka musi być otwarta z jedną lub więcej spośród następujących opcji (szczegółowe informacje na ten temat zawiera sekcja [“MQOPEN-obiekt otwarty” na stronie 742](#)):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc

Typ: MQMD-input/output

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu” na stronie 417](#).

Jeśli wartość *BufferLength* jest mniejsza niż długość komunikatu, *MsgDesc* jest wypełniona przez menedżer kolejek, niezależnie od tego, czy parametr MQGMO_ACCEPT_TRUNCATED_MSG jest określony w parametrze *GetMsgOpts* (patrz sekcja [MQGMO-opcje pola](#)).

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, zwrócony komunikat ma przedrostek MQMDE, który jest poprzedzony danymi komunikatu aplikacji, ale *tylko*, jeśli co najmniej jedno z pól w MQMDE ma wartość niedomyślną. Jeśli wszystkie pola w tabeli MQMDE mają wartości domyślne, pomijane jest MQMDE. Nazwa formatu MQFMT_MD_EXTENSION w polu *Format* w strukturze MQMD wskazuje, że jest obecna MQMDE.

Aplikacja nie musi udostępniać struktury MQMD, jeśli w polu *MsgHandle* dostarczony poprawny uchwyt komunikatu. Jeśli w tym polu nie zostanie podana żadna wartość, deskryptor komunikatu jest przyjmowany z deskryptora powiązanego z uchwytami komunikatów.

Jeśli aplikacja udostępnia uchwyt komunikatu, a nie strukturę MQMD, i określono parametr MQGMO_PROPERTIES_FORCE_MQRFH2, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_MD_ERROR. Wywołanie również kończy się niepowodzeniem z kodem przyczyny MQRC_MD_ERROR, jeśli aplikacja nie udostępnia struktury MQMD i określa wartość MQGMO_PROPERTIES_AS_Q_DEF, a atrybut kolejki produktu *PropertyControl* ma wartość MQPPROP_FORCE_MQRFH2.

Jeśli określono opcje zgodności, a deskryptor komunikatu powiązany z uchwytami komunikatu jest używany, pola wejściowe używane do dopasowywania pochodzą z uchwytu komunikatu.

GetMsgOpts (GetMsg)

Typ: MQGMO-input/output

Szczegółowe informacje na ten temat zawiera sekcja [“MQGMO-Opcje Get-message”](#) na stronie 367.

BufferLength

Typ: MQLONG-wejście

Jest to długość w bajtach obszaru *Buffer*. Podaj wartość zero dla komunikatów, które nie mają danych, lub jeśli komunikat ma zostać usunięty z kolejki, a dane zostały usunięte (w tym przypadku należy podać MQGMO_ACCEPT_TRUNCATED_MSG).

Uwaga: Długość najdłuższej wiadomości, którą można odczytać z kolejki, jest nadawana przez atrybut kolejki *MaxMsgLength*; patrz [“Atrybuty dla kolejek”](#) na stronie 845.

Buforuj

Typ: MQBYTEExBufferDługość-wyjście

Jest to obszar, w którym mają być zawarte dane komunikatu. Wyrównaj bufor na granicy, odpowiedni do charakteru danych w komunikacie. 4-bajtowe wyrównanie jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajтового.

Jeśli wartość *BufferLength* jest mniejsza niż długość komunikatu, to jak najwięcej komunikatu jest przenoszonych do produktu *Buffer*; Oznacza to, czy parametr MQGMO_ACCEPT_TRUNCATED_MSG jest określony w parametrze *GetMsgOpts* (więcej informacji na ten temat zawiera sekcja [MQGMO-opcje pola](#)).

Zestaw znaków i kodowanie danych w programie *Buffer* są nadawane przez pola *CodedCharSetId* i *Encoding* zwracane w parametrze *MsgDesc*. Jeśli wartości te różnią się od wartości wymaganych przez odbiorcę, odbiorca musi dokonać konwersji danych komunikatu aplikacji na wymagany zestaw znaków i kodowanie. Można użyć opcji MQGMO_CONVERT (jeśli jest to konieczne przy użyciu wyjścia napisanego przez użytkownika), aby przekształcić dane komunikatu. Szczegółowe informacje na temat tej opcji zawiera sekcja [“MQGMO-Opcje Get-message”](#) na stronie 367.

Uwaga: Wszystkie pozostałe parametry wywołania MQGET znajdują się w zestawie znaków i kodowaniu lokalnego menedżera kolejek (nadawanego przez atrybut menedżera kolejek produktu *CodedCharSetId* i atrybut MQENC_NATIVE).

Jeśli wywołanie nie powiedzie się, zawartość buforu może być nadal zmieniona.

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void: adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *BufferLength* ma wartość zero, *Buffer* nie jest przywołany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

DataLength

Typ: MQLONG-wyjście

Jest to długość (w bajtach) danych aplikacji w komunikacie. Jeśli wartość jest większa niż *BufferLength*, w parametrze *Buffer* zostaną zwrócone tylko *BufferLength* bajty (to znaczy, że komunikat jest obcinany). Jeśli wartość wynosi zero, komunikat nie zawiera danych aplikacji.

Jeśli wartość *BufferLength* jest mniejsza niż długość komunikatu, program *DataLength* jest nadal wypełniony przez menedżer kolejek, bez względu na to, czy parametr MQGMO_ACCEPT_TRUNCATED_MSG jest określony w parametrze *GetMsgOpts* (więcej informacji można znaleźć w sekcji [MQGMO-opcje pola](#)). Dzięki temu aplikacja może określić wielkość buforu wymaganego do obsługi danych komunikatu, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Jeśli jednak określono opcję MQGMO_CONVERT, a przekształcone dane komunikatu są zbyt długie, aby zmieściły się w programie *Buffer*, wartość zwrócona dla *DataLength* jest następująca:

- Długość danych *nieprzekształconych* dla formatów zdefiniowanych przez menedżera kolejek.

W takim przypadku, jeśli charakter danych powoduje jej rozszerzenie podczas konwersji, aplikacja musi przydzielić bufor większy niż wartość zwrócona przez menedżer kolejek dla produktu *DataLength*.

- Wartość zwracana przez wyjście konwersji danych dla formatów zdefiniowanych przez aplikację.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Podane kody przyczyn to te, które menedżer kolejek może zwrócić dla parametru *Reason*.

Jeśli aplikacja określa opcję MQGMO_CONVERT, a funkcja wyjścia napisana przez użytkownika jest wywoływana w celu przekształcenia niektórych lub wszystkich danych komunikatu, wyjście decyduje o tym, jaka wartość jest zwracana dla parametru *Reason*. W rezultacie wartości inne niż te, które zostały udokumentowane, są możliwe.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

MQRC_CONVERTED_STRING_TOO_DUZE

(2190, X'88E') Konwertowany łańcuch jest zbyt duży dla pola.

BŁĄD MQRC_DBCS_ERROR

(2150, X'866 ') Łańcuch DBCS nie jest poprawny.

BŁĄD FORMAT_MQRC_FORMAT_ERROR

(2110, X'83E') Format komunikatu nie jest poprawny.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') Grupa komunikatów nie została zakończona.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') Komunikat logiczny nie został zakończony.

MQRC_INCONSISTENT_CCIDS

(2243, X'8C3') Segmenty komunikatów mają różne identyfikatory CCSID.

MQRC_INCONSISTENT_ENCODINGS

(2244, X'8C4') Segmenty komunikatów mają różne kodowania.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') Niepoprawne użycie znacznika komunikatu.

MQRC_NO_MSG_LOCKED

(2209, X'8A1') Brak zablokowanego komunikatu.

MQRC_NOT_CONVERTED

(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

MQRC_OPTIONS_CHANGED

(nnnn, X'xxx ') Opcje, które musiały być spójne, zostały zmienione.

MQRC_PARTIALLY_PRZEKSZTAŁCONA

(2272, X'8E0') Dane komunikatu zostały częściowo przekształcone.

MQRC_SIGNAL_REQUEST_ACCEPTED

(2070, X'816 ') Nie zwrócono żadnego komunikatu (ale żądanie sygnału zostało zaakceptowane).

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861 ') Parametr buforu źródłowego jest niepoprawny.

MQRC_SOURCE_CCSD_ERROR, BŁĄD

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841 ') Zpakowane kodowanie dziesiętne w komunikacie nie zostało rozpoznane.

MQRC_SOURCE_FLOAT_ENC_ERROR, BŁĄD

(2114, X'842 ') Kodowanie zmiennopozycyjne w komunikacie nie zostało rozpoznane.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') Parametr długości źródła nie jest poprawny.

MQRC_TARGET_BUFFER_ERROR

(2146, X'862 ') Parametr buforu docelowego jest niepoprawny.

MQRC_TARGET_CCSD_ERROR

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

MQRC_TARGET_DECIMAL_ENC_ERROR,

(2117, X'845 ') Zpakowane-kodowanie dziesiętne określone przez odbiornik nierozpoznany.

MQRC_TARGET_FLOAT_ENC_ERROR, BŁĄD

(2118, X'846 ') Kodowanie zmiennopozycyjne określone przez odbiornik nie jest rozpoznawane.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') Zwrócona została obcięta wiadomość (przetwarzanie zostało zakończone).

Funkcja MQRC_TRUNCATED_MSG_FAILED

(2080, X'820 ') Zwrócona została obcięta wiadomość (przetwarzanie nie zostało zakończone).

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

Błąd MQRC_ADAPTER_CONV_LOAD_ERROR

(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BACKED_OUT

(2003, X'7D3') Wytworzona jednostka pracy.

MQRD_BUFFER_ERROR-BŁĄD

(2004, X'7D4') Parametr buforu nie jest poprawny.

MQRD_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRD_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRD_CF_NOT_AVAILABLE

(2345, X' 929 ') Obiekt sprzęgający nie jest dostępny.

MQRD_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X' 945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRD_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRD_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') Lista struktury narzędzia CF-nagłówek w użyciu.

MQRD_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRD_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRD_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRD_CONNECTION QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRD_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRD_CORREL_ID_ERROR (BŁĄD)

(2207, X'89F') Błąd korelacji identyfikatora.

Błąd MQRD_DATA_LENGTH_ERROR

(2010, X'7DA') Parametr długości danych nie jest poprawny.

MQRD_DB2_NOT_AVAILABLE

(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

MQRD_GET_INHIBITED

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

MQRD_GLOBAL_UOW_CONFLICT

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

BŁĄD MQRD_GMO_ERROR

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

MQRD_HANDLE_IN_USE_DLA_UOW

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

BŁĄD MQRD_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRD_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

MQRD_INCONSISTENT_BROWSE

(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRD_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRD_INVALID_MSG_UNDER_CURSOR

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT

(2352, X'930 ') Global unit of work conflicts with local unit of work.

BŁĄD MQRC_MATCH_OPTIONS_ERROR

(2247, X'8C7') Opcje zgodności nie są poprawne.

Błąd MQRC_MD_ERROR

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

BŁĄD MQRC_MSG_ID_ERROR

(2206, X'89E') Błąd identyfikatora komunikatu.

MQRC_MSG_SEQ_NUMBER_ERROR,

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

MQRC_MSG_TOKEN_ERROR

(2331, X'91B') Użycie znacznika komunikatu nie jest poprawne.

MQRC_NO_MSG_AVAILABLE

(2033, X'7F1') Brak dostępnego komunikatu.

MQRC_NO_MSG_UNDER_CURSOR

(2034, X'7F2') Przeglądaj kursor nie umieszczony na komunikacie.

MQRC_NOT_OPEN_FOR_BROWSE

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_INDEX_TYPE_ERROR

(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SECOND_MARK_NOT_ALLOWED

(2062, X'80E') Komunikat jest już oznaczony.

MQRC_SIGNAL_OUTSTANDING

(2069, X'815 ') Signal outstanding for this handle.

MQRC_SIGNAL1_ERROR

(2099, X'833 ') Pole sygnału nie jest poprawne.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

MQRC_SYNCPOINT_NOT_AVAILABLE

Obsługa punktów synchronizacji (2072, X'818 ') nie jest dostępna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

MQRC_WAIT_INTERVAL_ERROR

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

MQRC_WRONG_GMO_VERSION

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Pobrany komunikat jest zwykle usuwany z kolejki. To usunięcie może wystąpić jako część samego wywołania MQGET lub jako część punktu synchronizacji.

Opcje przeglądania to: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT i MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Jeśli opcja MQGMO_LOCK jest określona z jedną z opcji przeglądania, przejrany komunikat jest zablokowany, tak aby był widoczny tylko dla tego uchwytu.

Jeśli została określona opcja MQGMO_UNLOCK, poprzednio zablokowany komunikat jest odblokowany. W tym przypadku nie jest pobierany żaden komunikat, a parametry *MsgDesc*, *BufferLength*, *Buffer* i *DataLength* nie są sprawdzane ani zmieniane.

3. W przypadku aplikacji wywołujących wywołanie MQGET odczytany komunikat może zostać utracony, jeśli aplikacja zostanie zakończona nieprawidłowo lub połączenie zostanie zerwane podczas przetwarzania wywołania. Ten problem pojawia się, ponieważ odpowiednik działający na tej samej platformie co menedżer kolejek, który wydaje wywołanie MQGET w imieniu aplikacji, nie może wykryć utraty aplikacji do momentu, w którym odpowiedniki nie zostaną zwrócone do aplikacji, po usunięciu komunikatu z kolejki. Ten problem może wystąpić zarówno w przypadku komunikatów trwałych, jak i komunikatów nietrwałych.

Aby wyeliminować ryzyko utraty wiadomości w ten sposób, zawsze wczytywać wiadomości w obrębie jednostek pracy. Oznacza to, że określenie opcji MQGMO_SYNCPOINT w wywołaniu MQGET oraz użycie wywołań MQCMIT lub MQBACK w celu zatwierdzenia lub wycofania jednostki pracy po zakończeniu przetwarzania komunikatu. Jeśli określono parametr MQGMO_SYNCPOINT, a klient zakończy działanie w sposób nieprawidłowy lub połączenie zostanie zerwane, zastępcze wycofuje jednostkę pracy w menedżerze kolejek i komunikat zostanie przywrócony do kolejki. Więcej informacji na temat punktów synchronizacji można znaleźć w sekcji [Uwagi dotyczące punktu synchronizacji w aplikacjach produktu IBM MQ](#).

Taka sytuacja może mieć miejsce w przypadku klientów IBM MQ oraz aplikacji działających na tej samej platformie co menedżer kolejek.

4. Jeśli aplikacja umieszcza sekwencję komunikatów w konkretnym przypadku kolejce w ramach pojedynczej jednostki pracy, a następnie zatwierdza tę jednostkę pracy pomyślnie, komunikaty stają się dostępne do pobrania w następujący sposób:

- Jeśli kolejka jest kolejką *niewspółużytkowaną* (czyli kolejką lokalną), wszystkie komunikaty w obrębie jednostki pracy stają się dostępne w tym samym czasie.
- Jeśli kolejka jest kolejką *współużytkowaną*, komunikaty w obrębie jednostki pracy stają się dostępne w kolejności, w jakiej zostały umieszczone, ale nie wszystkie w tym samym czasie. Jeśli system jest mocno obciążony, to jest możliwe, aby pierwszy komunikat w jednostce pracy został pomyślnie pobrany, ale dla wywołania MQGET dla drugiego lub kolejnego komunikatu w jednostce pracy nie powiodło się wywołanie MQRC_NO_MSG_AVAILABLE. W przypadku wystąpienia tego problemu aplikacja musi czekać na krótką chwilę, a następnie ponowić próbę wykonania operacji.

5. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, kolejność tych komunikatów jest zachowywana, jeśli spełnione są określone warunki. Szczegółowe informacje na ten temat zawiera sekcja Uwagi dotyczące użycia MQPUT. Jeśli warunki są spełnione, komunikaty są prezentowane w aplikacji odbierającej w kolejności, w jakiej zostały wysłane, jeżeli:

- Tylko jeden odbiorca otrzymuje komunikaty z kolejki.

Jeśli istnieją dwie lub więcej aplikacji pobierających komunikaty z kolejki, muszą one uzgodnić z nadawcą mechanizm używany do identyfikowania komunikatów należących do sekwencji. Na przykład nadawca może ustawić wszystkie pola *CorrelId* w komunikatach w sekwencji do wartości, która była unikalna dla tej sekwencji komunikatów.

- Odbiornik nie zmienia celowo kolejności pobierania, na przykład przez określenie konkretnej *MsgId* lub *CorrelId*.

Jeśli aplikacja wysyłający komunikaty umieszcza komunikaty jako grupę komunikatów, komunikaty są prezentowane w aplikacji odbierającej w poprawnej kolejności, jeśli aplikacja odbierająca określa opcję MQGMO_LOGICAL_ORDER w wywołaniu MQGET. Więcej informacji na temat grup komunikatów zawiera sekcja:

- MQMD-pole MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Jeśli użytkownik otrzymuje komunikaty w grupie w punkcie synchronizacji, muszą upewnić się, że kompletna grupa jest przetwarzana przed podjęciem próby zakończenia transakcji.

6. Aplikacje muszą testować kod sprzężenia zwrotnego MQFB_QUIT w polu *Feedback* parametru *MsgDesc* i kończyć je, jeśli znajdują się w tej wartości. Więcej informacji na ten temat zawiera sekcja MQMD-informacja zwrotna.

7. Jeśli kolejka identyfikowana przez produkt *Hobj* została otwarta za pomocą opcji MQOO_SAVE_ALL_CONTEXT, a kod zakończenia z wywołania MQGET to MQCC_OK lub MQCC_WARNING, kontekst powiązany z uchwytami kolejki *Hobj* jest ustawiany na kontekst komunikatu, który został pobrany (chyba że ustawiona jest opcja MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT lub MQGMO_BROWSE_MSG_UNDER_CURSOR, w takim przypadku kontekst jest oznaczony jako niedostępny).

Zapisanego kontekstu można użyć w kolejnych wywołaniach MQPUT lub MQPUT1, podając opcje MQPMO_PASS_IDENTITY_CONTEXT lub MQPMO_PASS_ALL_CONTEXT. Umożliwia to przesyłanie kontekstu komunikatu, który ma zostać przesłany w całości lub w części, do innego komunikatu (na przykład, gdy komunikat jest przekazywany do innej kolejki). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

8. Jeśli w parametrze *GetMsgOpts* zostanie włączona opcja MQGMO_CONVERT, dane komunikatu aplikacji zostaną przekonwertowane na reprezentację żądaną przez aplikację odbierającą, zanim dane zostaną umieszczone w parametrze *Buffer*:

- Pole *Format* znajdujące się w informacjach sterujących w komunikacie identyfikuje strukturę danych aplikacji, a pola *CodedCharSetId* i *Encoding* w informacjach sterujących w komunikacie określają jego identyfikator i kodowanie zestawu znaków.
- Aplikacja wywołująca wywołanie MQGET określa w polach *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* identyfikator zestawu znaków i kodowanie, do którego mają zostać przekształcone dane komunikatu aplikacji.

Gdy konieczna jest konwersja danych komunikatu, konwersja jest wykonywana przez sam menedżer kolejek lub przez wyjście napisane przez użytkownika, w zależności od wartości pola *Format* w informacjach sterujących w komunikacie:

- Następujące nazwy formatów są formatami przekształcanymi przez menedżer kolejek. Te formaty są nazywane formatami wbudowanymi:
 - ADMINISTRATOR MQFMT_ADMIN
 - MQFMT_CICS (tylko z/OS)
 - MQFMT_COMMAND_1
 - MQFMT_COMMAND_2
 - MQFMT_DEAD_LETTER_HEADER
 - MQFMT_DIST_HEADER
 - MQFMT_EVENT, wersja 1
 - MQFMT_EVENT, wersja 2 (tylko 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 (tylko z/OS)
 - MQFMT_XMIT_Q_HEADER
- Nazwa formatu MQFMT_NONE to wartość specjalna, która wskazuje, że charakter danych w komunikacie nie jest zdefiniowany. W związku z tym menedżer kolejek nie próbuje konwersji, gdy komunikat jest pobierany z kolejki.

Uwaga: Jeśli w wywołaniu MQGET określono wartość MQGMO_CONVERT dla komunikatu, który ma nazwę formatu MQFMT_NONE, a zestaw znaków lub kodowanie komunikatu różni się od wartości określonej w parametrze *MsgDesc*, to komunikat jest zwracany w parametrze *Buffer* (nie przyjmując innych błędów), ale wywołanie kończy się kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_FORMAT_ERROR.

Parametru MQFMT_NONE można użyć, gdy rodzaj danych komunikatu oznacza, że nie wymaga konwersji, lub gdy aplikacje wysyłający i odbierający uzgodniły między sobą formularz, w którym mają zostać wysłane dane komunikatu.

- Wszystkie inne nazwy formatu przekazują komunikat do programu zewnętrznego, który został napisany przez użytkownika w celu konwersji. Wyjście ma taką samą nazwę, jak format, poza dodatkami specyficznymi dla środowiska. Nazwy formatów podane przez użytkownika nie mogą rozpoczynać się od liter IBM MQ.

Szczegółowe informacje na temat wyjścia konwersji danych znajdują się w sekcji [“Konwersja danych” na stronie 919](#).

Dane użytkownika w komunikacie mogą być konwertowane między dowolnymi obsługiwanyymi zestawami znaków i kodowaniami. Należy jednak pamiętać, że jeśli komunikat zawiera co najmniej jedną strukturę nagłówka IBM MQ, nie można przekształcić komunikatu z zestawu znaków lub zestawu znaków zawierającego znaki dwubajtowe lub wielobajtowe dla dowolnych znaków, które są poprawne w nazwach kolejek. Kod przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR powoduje, że próba ta jest wykonywana, a komunikat jest zwracany bez konwersji. Zestaw znaków Unicode UCS-2 jest przykładem takiego zestawu znaków.

Po powrocie z wywołania MQGET następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:

- MQRC_NONE

Następujący kod przyczyny wskazuje, że komunikat *mógł* zostać pomyślnie przekształcony. Aplikacja musi sprawdzić pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc*, aby dowiedzieć się, jakie są:

- MQRC_TRUNCATED_MSG_ACCEPTED

Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Uwaga: Interpretacja tego kodu przyczyny jest prawdziwa dla konwersji wykonywanych przez wypisane przez użytkownika wyjście *tylko*, jeśli wyjście jest zgodne z wytycznymi przetwarzania opisanymi w „Konwersja danych” na stronie 919.

9. Jeśli do pobierania komunikatów używany jest interfejs obiektowy, można zdecydować, aby nie określać buforu, w którym mają być przechowywane dane komunikatu dla wywołania MQGET. Jednak w wersjach produktu IBM MQ w wersjach wcześniejszych niż 7 możliwe było niepowodzenie operacji MQGET z kodem przyczyny MQRC_CONVERTED_MSG_TO_BIG, nawet jeśli nie określono buforu. W produkcie IBM MQ 7 po otrzymaniu komunikatu za pomocą aplikacji obiektowej bez ograniczenia wielkości buforu komunikatów odbierania aplikacja nie kończy się niepowodzeniem z opcją MQRC_CONVERTED_MSG_TOO_BIG, a następnie otrzymuje przekształcony komunikat. Jest to prawda w następujących środowiskach:

- .NET, w tym w pełni zarządzane aplikacje
- C++
- Java (IBM MQ classes for Java)

Uwaga: W przypadku wszystkich klientów, jeśli wartość parametru *sharingConversations* wynosi zero, kanał działa tak, jak przed IBM WebSphere MQ 7.0, a obsługa komunikatów wycofuje się do zachowania Wersja 6. W takiej sytuacji, jeśli bufor jest zbyt mały, aby otrzymać przekształcony komunikat, zwracany jest komunikat o nieprzekształconej wersji, o kodzie przyczyny MQRC_CONVERTED_MSG_TOO_BIG. Więcej informacji na temat produktu *sharingConversations* zawiera sekcja Używanie konwersacji współużytkowanych w aplikacji klienckiej.

10. W przypadku wbudowanych formatów menedżer kolejek może wykonać *domyślną konwersję* łańcuchów znaków w komunikacie, gdy określona jest opcja MQGMO_CONVERT. Domyślna konwersja umożliwia menedżerowi kolejek korzystanie z domyślnego zestawu znaków określonego przez instalację, który przybliża rzeczywisty zestaw znaków podczas przekształcania danych łańcuchowych. W rezultacie wywołanie MQGET może zakończyć się pomyślnie kodem zakończenia MQCC_OK, a nie zakończyć się poprawką MQCC_WARNING i kodem przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji danych łańcuchowych jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Aby tego uniknąć, należy użyć znaków w łańcuchu, które są wspólne zarówno dla rzeczywistego zestawu znaków, jak i domyślnego zestawu znaków.

Domyślna konwersja ma zastosowanie zarówno do danych komunikatu aplikacji, jak i do pól znakowych w strukturach MQMD i MQMDE:

- Domyślna konwersja danych komunikatu aplikacji jest wykonywana tylko wtedy, gdy *wszystkie* są prawdziwe:

- Aplikacja określa wartość MQGMO_CONVERT.
- Komunikat zawiera dane, które muszą zostać przekształcone z lub do zestawu znaków, który nie jest obsługiwany.
- Domyślna konwersja została włączona podczas instalowania lub restartowania menedżera kolejek.
- W razie potrzeby domyślna konwersja pól znakowych w strukturach MQMD i MQMDE, jeśli dla menedżera kolejek włączona jest konwersja domyślna. Konwersja jest wykonywana nawet wtedy, gdy opcja MQGMO_CONVERT nie jest określona przez aplikację w wywołaniu MQGET.

11. W przypadku języka programowania Visual Basic, zastosowanie mają następujące punkty:

- Jeśli wielkość parametru *Buffer* jest mniejsza niż długość określona przez parametr *BufferLength*, wywołanie nie powiedzie się i zostanie podany kod przyczyny MQRC_STORAGE_NOT_AVAILABLE.
- Parametr *Buffer* jest zadeklarowany jako typ String. Jeśli dane, które mają zostać pobrane z kolejki, nie są typu String, należy użyć Wywołanie MQGETAny w miejscu wywołania MQGET.

Wywołanie MQGETAny ma takie same parametry jak wywołanie MQGET, z wyjątkiem tego, że parametr *Buffer* jest zadeklarowany jako typ Any, co pozwala na pobranie dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić *Buffer*, aby upewnić się, że wielkość ta wynosi co najmniej *BufferLength* bajtów.

12. Nie wszystkie opcje MQGET są obsługiwane, jeśli funkcja odczytu z wyprzedzeniem jest włączona. W poniższej tabeli wskazano, które opcje są dozwolone oraz czy mogą być zmieniane między wywołaniami MQGET.

Tabela 105. Opcje MQGET są dozwolone, gdy opcja odczytu z wyprzedzeniem jest włączona			
	Dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona i może być zmieniana między wywołaniami MQGET	Dozwolone, jeśli funkcja odczytu z wyprzedzeniem jest włączona, ale nie może być zmieniana między wywołaniami MQGET ^a	Opcje MQGET, które nie są dozwolone, gdy funkcja odczytu z wyprzedzeniem jest włączona, ^b
Wartości MQGET MD	MsgId ^c CorrelId ^c	Kodowanie CodedCharSetId	
Opcje MQGMO MQGET	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 Komunikat MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER _CURSOR ^d Blokada MQGMO_LOCK MQGMO_UNLOCK
Wartości MQGMO		MsgHandle	

- Jeśli te opcje zostaną zmienione między wywołaniami MQGET, zostanie zwrócony kod przyczyny MQRC_OPTIONS_CHANGED.
- Jeśli te opcje zostaną podane podczas pierwszego wywołania MQGET, odczyt z wyprzedzeniem zostanie wyłączony. Jeśli te opcje zostaną podane w kolejnym wywołaniu MQGET, zostanie zwrócony kod przyczyny MQRC_OPTIONS_ERROR.
- Aplikacje klienckie muszą uwzględniać fakt, że jeśli wartości MsgId i CorrelId zostały zmienione między wywołaniami MQGET, komunikaty z poprzednimi wartościami mogły już zostać wysłane do klienta i pozostają w buforze odczytu z wyprzedzeniem na kliencie, dopóki nie zostaną wykorzystane (lub automatycznie usunięte).
- Pierwsze wywołanie MQGET określa, czy komunikaty mają być przeglądane lub pobierane z kolejki, gdy włączony jest odczyt z wyprzedzeniem. Jeśli w aplikacji zostanie podjęta próba użycia zarówno operacji przeglądania, jak i pobierania, zostanie zwrócony kod przyczyny MQRC_OPTIONS_CHANGED.

- e. Opcja MQGMO_MSG_UNDER_CURSOR nie jest dostępna, jeśli włączony jest odczyt z wyprzedzeniem. Komunikaty można przeglądać albo odbierać, gdy włączony jest odczyt z wyprzedzeniem. Nie można jednak jednocześnie korzystać z obu tych funkcji.
- 13. Aplikacje mogą destrukcyjnie uzyskać niezatwierdzone komunikaty tylko wtedy, gdy te komunikaty są umieszczane w tej samej lokalnej jednostce pracy, co element get. Aplikacje nie mogą uzyskać niezatwierdzonych komunikatów nieniszczących.
- 14. Komunikaty pod kursorem przeglądania mogą być pobierane w jednostce pracy. Nie jest możliwe pobranie niezatwierdzonej wiadomości w ten sposób.

Wywołanie C

```
MQGET (Hconn, Hobj, &MsgDesc, &GetMsgOpts, BufferLength, Buffer,
       &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQMD     MsgDesc;         /* Message descriptor */
MQGMO    GetMsgOpts;      /* Options that control the action of MQGET */
MQLONG   BufferLength;     /* Length in bytes of the Buffer area */
MQBYTE   Buffer[n];        /* Area to contain the message data */
MQLONG   DataLength;      /* Length of the message */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
                  BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ           PIC S9(9) BINARY.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Options that control the action of MQGET
01 GETMSGOPTS.
   COPY 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.
```

Wywołanie PL/I

```
call MQGET (Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
            DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl MsgDesc        like MQMD;    /* Message descriptor */
dcl GetMsgOpts     like MQGMO;    /* Options that control the action of
                                   MQGET */
dcl BufferLength    fixed bin(31); /* Length in bytes of the Buffer
                                   area */
dcl Buffer          char(n);       /* Area to contain the message data */
dcl DataLength      fixed bin(31); /* Length of the message */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */

```

Wywołanie High Level Assembler

```

CALL MQGET, (HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
            BUFFER, DATALENGTH, COMPCODE, REASON)

```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
GETMSGOPTS	CMQGMOA	,	Options that control the action of MQGET
BUFFERLENGTH	DS	F	Length in bytes of the BUFFER area
BUFFER	DS	CL(n)	Area to contain the message data
DATALENGTH	DS	F	Length of the message
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```

MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason

```

Zadeklaruj parametry w następujący sposób:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim MsgDesc	As MQMD	'Message descriptor'
Dim GetMsgOpts	As MQGMO	'Options that control the action of MQGET'
Dim BufferLength	As Long	'Length in bytes of the Buffer area'
Dim Buffer	As String	'Area to contain the message data'
Dim DataLength	As Long	'Length of the message'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQINQ-zapytanie o atrybuty obiektu

Wywołanie funkcji MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Poprawne są następujące typy obiektów:

- Menedżer kolejek
- Kolejka
- Lista nazw
- Definicja procesu

Składnia

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN -wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN i podać następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Hobj

Typ: MQHOBJ -wejście

Ten uchwyt reprezentuje obiekt (dowolnego typu) z wymaganymi atrybutami. Uchwyt musi zostać zwrócony przez poprzednie wywołanie MQOPEN, które określiło opcję MQOO_INQUIRE.

SelectorCount

Typ: MQLONG -wejście

Jest to liczba selektorów, które są dostarczane w macierzy *Selectors*. Jest to liczba atrybutów, które mają zostać zwrócone. Wartość zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

Selektory

Typ: MQLONG x *SelectorCount* -wejście

Jest to tablica selektorów atrybutów *SelectorCount*; każdy selektor identyfikuje atrybut (liczba całkowita lub znak) z wymaganą wartością.

Każdy selektor musi być poprawny dla typu obiektu reprezentowanego przez produkt *Hobj*.

W przeciwnym razie wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_SELECTOR_ERROR.

W specjalnym przypadku kolejek:

- Jeśli selektor nie jest poprawny dla kolejek dowolnego typu, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_SELECTOR_ERROR.
- Jeśli selektor ma zastosowanie tylko do kolejek typów innych niż typ obiektu, wywołanie powiedzie się z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_SELECTOR_NOT_FOR_TYPE.
- Jeśli zapytanie o kolejkę jest kolejką klastra, poprawne selektory zależą od sposobu, w jaki kolejka została rozstrzygnięta. Więcej informacji na ten temat zawiera sekcja ["Użycie notatek" na stronie 729](#).

Selektory można określać w dowolnej kolejności. Wartości atrybutów, które odpowiadają selektorom atrybutu liczby całkowitej (selektory MQIA_*), są zwracane w produkcie *IntAttrs* w tej samej kolejności, w jakiej te selektory występują w produkcie *Selectors*. Wartości atrybutów, które odpowiadają selektorom atrybutów znakowych (selektory MQCA_*), są zwracane w produkcie *CharAttrs* w tej samej kolejności, w jakiej występują te selektory. Selektory MQIA_* można przeplatać się z selektorami MQCA_*. Ważne jest tylko to, że kolejność względna w poszczególnych typach jest istotna.

Uwaga:

1. Selektory atrybutów całkowitoliczbowych i atrybutów znakowych są przydzielane w dwóch różnych zakresach; selektory MQIA_* znajdują się w zakresie od MQIA_FIRST do MQIA_LAST, a selektory MQCA_* w zakresie od MQCA_FIRST do MQCA_LAST.

Dla każdego zakresu stałe MQIA_LAST_USED i MQCA_LAST_USED definiują najwyższą wartość akceptowania przez menedżer kolejek.

2. Jeśli wszystkie selektory MQIA_* występują jako pierwsze, te same numery elementów mogą być używane do adresowania odpowiednich elementów w macierzach *Selectors* i *IntAttrs*.
3. If the *SelectorCount* parameter is zero, *Selectors* is not referred to. W tym przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 assembler może mieć wartość NULL.

Atrybuty, które można uzyskać do zapytania, są wymienione w poniższych tabelach. W przypadku selektorów MQCA_* stała, która definiuje długość łańcucha wynikowego w bajtach w programie *CharAttrs*, jest podana w nawiasach.

Tabele, które śledzą listę selektorów, według obiektu, w kolejności alfabetycznej, są następujące:

- Selektory atrybutów [Tabela 106 na stronie 716 MQINQ dla kolejek](#)
- Selektory atrybutów [Tabela 107 na stronie 719 MQINQ dla list nazw](#)
- Selektory atrybutów [Tabela 108 na stronie 719 MQINQ dla definicji procesów](#)
- Selektory atrybutów [Tabela 109 na stronie 720 MQINQ dla menedżera kolejek](#)

Wszystkie selektory są obsługiwane na wszystkich platformach IBM MQ, z wyjątkiem sytuacji, gdy jest to wskazane w kolumnie **Uwaga** w następujący sposób:

NIEz/OS

Obsługiwane na wszystkich platformach **z wyjątkiem** z/OS

z/OS

Obsługiwane **tylko** w systemie z/OS

Tabela 106. MQINQ selektory atrybutów dla kolejek			
Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Godzina ostatniej zmiany	
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa nadmiernej liczby wycofanych komunikatów	
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki, która jest tłumaczona na alias	
MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nazwa struktury narzędzia CF	z/OS
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nazwa kanału nadawczego klastra, który używa tej kolejki jako kolejki transmisji.	
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nazwa klastra	
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Lista nazw klastrów	
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Data utworzenia kolejki	

Tabela 106. MQINQ selektory atrybutów dla kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Czas utworzenia kolejki	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	Atrybut niestandardowy dla nowych funkcji	
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki inicjacji	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nazwa definicji procesu	
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Opis kolejki	
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki	
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nazwa zdalnego menedżera kolejek	
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki zdalnej, która jest znana w zdalnym menedżerze kolejek	
MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nazwa klasy pamięci masowej	z/OS
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Dane wyzwalacza	
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki transmisji	
MQIA_ACCOUNTING_Q	MQLONG	Steruje gromadzeniem danych rozliczeniowych dla kolejki	NIEz/OS
MQIA_BACKOUT_THRESHOLD	MQLONG	Próg wycofania	
MQIA_CLWL_Q_PRIORITY	MQLONG	Priorytet kolejki	
MQIA_CLWL_Q_RANK	MQLONG	Ranga kolejki	
MQIA_CLWL_USEQ	MQLONG	Użyj kolejek zdalnych	
MQIA_CURRENT_Q_DEPTH	MQLONG	Liczba komunikatów w kolejce	
MQIA_DEF_BIND	MQLONG	Domyślne łączenie	
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Domyślna opcja open-for-input	
MQIA_DEF_PERSISTENCE	MQLONG	Domyślna trwałość komunikatu	
MQIA_DEF_PRIORITY	MQLONG	Domyślny priorytet komunikatu	
MQIA_DEFINITION_TYPE	MQLONG	Typ definicji kolejki.	
MQIA_DIST_LISTS	MQLONG	Obsługa listy dystrybucyjnej	NIEz/OS
MQIA_HARDEN_GET_BACKOUT	MQLONG	Czy ma być wycofana liczba wycofań	
MQIA_INDEX_TYPE	MQLONG	Typ indeksu utrzymanego dla kolejki	z/OS
MQIA_INHIBIT_GET	MQLONG	Czy dozwolone są operacje pobierania	
MQIA_INHIBIT_PUT	MQLONG	Czy dozwolone są operacje put	
MQIA_MAX_MSG_LENGTH	MQLONG	Maksymalna długość komunikatu	

Tabela 106. MQINQ selektory atrybutów dla kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_MAX_Q_DEPTH	MQLONG	Maksymalna liczba komunikatów dozwolonych w kolejce	
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Określa, czy priorytet komunikatu ma znaczenie	
MQIA_NPM_CLASS	MQLONG	Poziom niezawodności komunikatów nietrwałych	
MQIA_OPEN_INPUT_COUNT	MQLONG	Liczba wywołań MQOPEN , które mają otwartą kolejkę dla wejścia	
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Liczba wywołań MQOPEN , które mają otwartą kolejkę dla danych wyjściowych	
MQIA_PROPERTY_CONTROL	MQLONG	Atrybut elementu sterującego właściwości	
MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń wysokiego zapętnienia kolejki	NIEz/OS
MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Górny limit głębokości kolejki	NIEz/OS
MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń o niskiej głębokości kolejki	NIEz/OS
MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Niski limit głębokości kolejki	NIEz/OS
MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń maksymalnej głębokości kolejki	NIEz/OS
MQIA_Q_SERVICE_INTERVAL	MQLONG	Limit czasu dla usługi kolejki	NIEz/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń przedziału czasu usługi kolejki	NIEz/OS
MQIA_Q_TYPE	MQLONG	Typ kolejki	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS
MQIA_RETENTION_INTERVAL	MQLONG	Interwał czasu przechowywania kolejki	
MQIA_SCOPE	MQLONG	Zasięg definicji kolejki	NIEz/OS
MQIA_SHAREABILITY	MQLONG	Określa, czy kolejka może być współużytkowana dla danych wejściowych	
MQIA_STATISTICS_Q	MQLONG	Steruje gromadzeniem danych statystycznych dla kolejki	NIEz/OS
MQIA_TRIGGER_CONTROL	MQLONG	Kontrola wyzwalacza	
MQIA_TRIGGER_DEPTH	MQLONG	Wyzwalacz uruchamiany zapętnieniem	

Tabela 106. MQINQ selektory atrybutów dla kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Próg priorytetu komunikatu dla wyzwalacza.	
MQIA_TRIGGER_TYPE	MQLONG	Typ wyzwalacza	
MQIA_USAGE	MQLONG	Użycie	

Tabela 107. Selektory atrybutów MQINQ dla list nazw			
Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Godzina ostatniej zmiany	
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Opis listy nazw	
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nazwa obiektu listy nazw	
MQIA_NAMELIST_TYPE	MQLONG	Typ listy nazw	z/OS
MQCA_NAMES	MQ_Q_NAME_LENGTH x Number of names in the list	Nazwy na liście nazw	
MQIA_NAME_COUNT	MQLONG	Liczba nazw na liście nazw	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS

Tabela 108. Selektory atrybutów MQINQ dla definicji procesów			
Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Godzina ostatniej zmiany	
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identyfikator aplikacji	
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Dane środowiska	
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Opis definicji procesu	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nazwa definicji procesu	
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Dane użytkownika	
MQIA_APPL_TYPE	MQLONG	Typ aplikacji	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek			
Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Godzina ostatniej zmiany	
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nazwa wyjścia automatycznej definicji kanału	
MQCA_CHINIT_SERVICE_PARM		Zarezerwowane do użycia przez produkt IBM	
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Dane przekazane do wyjścia obciążenia klastra	
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nazwa wyjścia obciążenia klastra	
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki wejściowej komend systemowych	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	Atrybut niestandardowy dla nowych funkcji	
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki niedostarczonych komunikatów	
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Domyślna nazwa kolejki transmisji	
MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Nazwa grupy dla obiektu nasłuchiwania TCP, który obsługuje transmisje przychodzące dla grupy współużytkowania kolejki, do której ma zostać przyłączone połączenie. Ta nazwa ma zastosowanie w przypadku korzystania z usług Active Domain Name Services programu Workload Manager.	z/OS
MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identyfikator użytkownika kolejkowania wewnątrz grupy	z/OS
MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Opis powiązanej instalacji	Nie z/OS · NieIBM i
MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nazwa instalacji powiązanej z menedżerem kolejek	Nie z/OS · NieIBM i
MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Ścieżka, w której jest zainstalowany powiązany IBM MQ	Nie z/OS · NieIBM i

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Nazwa ogólnej jednostki logicznej dla programu nastuchującego LU 6.2 obsługującego transmisje przychodzące dla grupy współużytkowania kolejki, która ma być używana	z/OS
MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 . Tę nazwę należy ustawić na tę samą jednostkę logiczną, która jest używana przez proces nastuchujący na potrzeby transmisji danych przychodzących.	z/OS
MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Suffix of the SYS1 . PARMLIB member APPCPM <i>xx</i> , that nominates the LUADD for this channel initiator	z/OS
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nazwa hierarchicznie połączonego menedżera kolejek, który jest nominowany jako element nadrzędny tego menedżera kolejek.	
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Opis menedżera kolejek	
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGTH	Identyfikator menedżera kolejek (H)	
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nazwa lokalnego menedżera kolejek	
MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nazwa grupy współużytkowania kolejki	z/OS
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nazwa klastra, dla którego menedżer kolejek udostępnia usługi repozytorium	
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których menedżer kolejek udostępnia usługi repozytorium	
MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Nazwa systemu TCP/IP, który jest używany.	z/OS
MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Nadpisz ustawienia rozliczania	NIEz/OS
MQIA_ACCOUNTING_INTERVAL	MQLONG	Jak często zapisywać pośrednie rekordy rozliczeniowe	NIEz/OS
MQIA_ACCOUNTING_MQI	MQLONG	Steruje gromadzeniem informacji rozliczeniowych dla danych MQI	NIEz/OS
MQIA_ACCOUNTING_Q	MQLONG	Steruje gromadzeniem informacji rozliczeniowych dla kolejek	NIEz/OS

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_ACTIVE_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie	z/OS
MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementy, które są sprawdzane w celu określenia, czy należy adoptować agenta MCA. Sprawdzenie jest wykonywane po wykryciu nowego kanału danych przychodzących o tej samej nazwie co agent MCA, który jest już aktywny.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Ilość czasu (w sekundach), przez jaki nowy kanał oczekuje na zakończenie osieroconego kanału	NIEz/OS
MQIA_ADOPTNEWMCA_TYPE	MQLONG	Określa, czy automatycznie restartować osierocone instancje agenta MCA określonego typu kanału w przypadku wykrycia nowego żądania kanału przychodzącego zgodnego z parametrami AdoptNewMCACheck .	z/OS
MQIA_AUTHORITY_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń uprawnień	NIEz/OS
MQIA_BRIDGE_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń mostu IMS	z/OS
MQIA_CHANNEL_AUTO_DEF	MQLONG	Atrybut elementu sterującego dla definicji kanału automatycznego	NIEz/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń automatycznej definicji kanału	NIEz/OS
MQIA_CHANNEL_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń kanału	
MQIA_CHINIT_ADAPTERS	MQLONG	Liczba podzadań adaptera, które mają być używane do przetwarzania wywołań IBM MQ	z/OS
MQIA_CHINIT_DISPATCHERS	MQLONG	Liczba programów rozsyłających, które mają zostać użyte dla inicjatora kanału	z/OS
MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Określa, czy śledzenie inicjatora kanału ma być uruchamiane automatycznie	z/OS
MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Wielkość obszaru danych śledzenia (w MB) inicjatora kanału	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Długość obciążenia klastra.	
MQIA_CLWL_MRU_CHANNELS	MQLONG	Liczba ostatnio używanych kanałów dla równoważenia obciążenia klastra	

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_CLWL_USEQ	MQLONG	Użyj kolejek zdalnych	
MQIA_CODED_CHAR_SET_ID	MQLONG	Identyfikator kodowanego zestawu znaków	
MQIA_COMMAND_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń komendy	
MQIA_COMMAND_LEVEL	MQLONG	Poziom komend obsługiwany przez menedżer kolejek	
MQIA_CONFIGURATION_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń konfiguracji	NIEz /OS
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Domyślny typ kolejki transmisji, która ma być używana w przypadku kanałów nadawczych klastra.	
MQIA_DIST_LISTS	MQLONG	Obsługa listy dystrybucyjnej	NIEz /OS
MQIA_DNS_WLM	MQLONG	Określa, czy obiekt nasłuchiwania TCP obsługujący transmisje danych przychodzących dla grupy współużytkowania kolejki jest rejestrowany w programie Workload Manager for Dynamic Domain Name Services.	z/OS
MQIA_EXPIRY_INTERVAL	MQLONG	Odstęp czasu między kolejnymi skanowaniami komunikatów, które	z/OS
MQIA_GROUP_UR	MQLONG	Atrybut elementu sterującego dla tego, czy dla tego menedżera kolejek włączone są jednostki odzyskiwania grupy. Dyspozycja jednostki odzyskiwania grupy jest dostępna tylko wtedy, gdy menedżer kolejek należy do grupy współużytkowania kolejki.	z/OS
MQIA_IGQ_PUT_AUTHORITY	MQLONG	Uprawnienie do umieszczania w kolejkach wewnątrz grupy	z/OS
MQIA_INHIBIT_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń zablokowanej	NIEz /OS
MQIA_INTRA_GROUP_queuing	MQLONG	Obsługa kolejkowania wewnątrz grupy	z/OS
MQIA_LISTENER_TIMER	MQLONG	Odstęp czasu (w sekundach) między kolejnymi próbami zrestartowania obiektu nasłuchiwania przez program IBM MQ , jeśli komunikacja APPC lub TCP/IP nie powiodła się.	z/OS
MQIA_LOCAL_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń lokalnych	NIEz /OS

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_LOGGER_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń zablokowanej	NIEz/OS
MQIA_LU62_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być bieżące lub klienci, które mogą być podłączone, za pomocą protokołu transmisji LU 6.2	z/OS
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Odstęp czasu (w milisekundach), po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania.  Ostrzeżenie: Nie należy ustawiać tej wartości poniżej wartości domyślnej 5000.	
MQIA_MAX_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być aktualne (w tym kanały połączenia z serwerem z połączonymi klientami)	z/OS
MQIA_MAX_HANDLES	MQLONG	Maksymalna liczba uchwytów	
MQIA_MAX_MSG_LENGTH	MQLONG	Maksymalna długość komunikatu	
MQIA_MAX_PRIORITY	MQLONG	Maksymalny priorytet	
MQIA_MAX_UNCOMMITTED_MESSAGES	MQLONG	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy	
MQIA_OUTBOUND_PORT_MAX	MQLONG	Program MQIA_OUTBOUND_PORT_MIN definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.	z/OS
MQIA_OUTBOUND_PORT_MIN	MQLONG	Program MQIA_OUTBOUND_PORT_MAX definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.	z/OS
MQIA_PERFORMANCE_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń wydajności	NIEz/OS
MQIA_PLATFORM	MQLONG	Platforma, na której znajduje się menedżer kolejek	
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Wskazuje, czy funkcje zabezpieczeń produktu IBM MQ Advanced Message Security są dostępne dla menedżera kolejek.	
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	Liczba prób ponownego przetworzenia komunikatu komendy zakończonej niepowodzeniem w punkcie synchronizacji	

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_PUBSUB_MODE	MQLONG	Określa, czy działa mechanizm publikowania/subskrypcji, a także umieszczony w kolejce interfejs publikowania/subskrypcji. Aplikacje do publikowania lub subskrybowania przy użyciu interfejsu programistycznego aplikacji wymagają mechanizmu publikowania/subskrypcji. Kolejki monitorowane przez interfejs w kolejce publikowania/subskrypcji wymagają, aby interfejs publikowania/subskrybowania w kolejce był uruchomiony.	
MQIA_PUBSUB_NP_MSG	MQLONG	Informacja o tym, czy usunąć (lub zachować) niedostarczone komunikaty wejściowe	
MQIA_PUBSUB_NP_RESP	MQLONG	Steruje zachowaniem niedostarczonych komunikatów odpowiedzi	
MQIA_PUBSUB_SYNC_PT	MQLONG	Określa, czy tylko trwałe (lub wszystkie) komunikaty są przetwarzane w punkcie synchronizacji	
MQIA_QMGR_CFCONLOS	MQLONG	Określa działanie, które ma zostać podjęte, gdy menedżer kolejek utraci połączenie ze strukturą administracyjną lub dowolnymi strukturami systemu CF z CFCONLOS ustawionym na wartość ASQMGR .	z/OS
MQIA_RECEIVE_TIMEOUT	MQLONG	W przybliżeniu, jak długo kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od swojego partnera, przed powrotem do stanu nieaktywnego. Wartość jest liczbowa, która jest kwalifikowana przez produkt MQIA_RECEIVE_TIMEOUT_TYPE.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Minimalny czas, przez jaki kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od partnera, przed powrotem do stanu nieaktywnego	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	W przybliżeniu, jak długo kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od swojego partnera, przed powrotem do stanu nieaktywnego. MQIA_RECEIVE_TIMEOUT_TYPE to kwalifikator zastosowany do MQIA_RECEIVE_TIMEOUT.	z/OS

Tabela 109. MQINQ selektory atrybutów dla menedżera kolejek (kontynuacja)			
Selektor	Długość pola	Opis	Uwaga
MQIA_REMOTE_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń zdalnych	NIEz /OS
MQIA_SECURITY_CASE	MQLONG	Przypadek profili zabezpieczeń	z/OS
MQIA_SSL_EVENT	MQLONG	Atrybut elementu sterującego dla zdarzeń kanału	
MQIA_SSL_FIPS_REQUIRED	MQLONG	Używaj tylko algorytmów z certyfikatem FIPS dla kryptografii	
MQIA_SSL_RESET_COUNT	MQLONG	Licznik zerowania klucza SSL	
MQIA_START_STOP_EVENT	MQLONG	Atrybut elementu sterującego uruchamiania zdarzeń zatrzymania	NIEz /OS
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Steruje gromadzeniem informacji o monitorowaniu statystyk dla kanałów nadajnika klastrów	
MQIA_STATISTICS_CHANNEL	MQLONG	Steruje gromadzeniem danych statystycznych dla kanałów	
MQIA_STATISTICS_INTERVAL	MQLONG	Jak często zapisywać dane monitorowania statystyk	NIEz /OS
MQIA_STATISTICS_MQI	MQLONG	Steruje gromadzeniem informacji o monitorowaniu statystyk dla menedżera kolejek	NIEz /OS
MQIA_STATISTICS_Q	MQLONG	Steruje gromadzeniem danych statystycznych dla kolejek	NIEz /OS
MQIA_SYNCPOINT	MQLONG	dostępność punktu synchronizacji	
MQIA_TCP_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być aktualne, lub klientów, które mogą być podłączone, za pomocą protokołu transmisji TCP/IP	z/OS
MQIA_TCP_KEEP_ALIVE	MQLONG	Określa, czy użyć narzędzia TCP KEEPALIVE do sprawdzenia, czy drugi koniec połączenia jest nadal dostępny	z/OS
MQIA_TCP_STACK_TYPE	MQLONG	Określa, czy inicjator kanału może używać tylko przestrzeni adresowej TCP/IP określonej w nazwie TCPNAME, czy też może być opcjonalnie powiązany z dowolnym wybranym adresem TCP/IP	z/OS
MQIA_TRACE_ROUTE_RECORDING	MQLONG	Steruje rejestrowaniem informacji o trasie śledzenia	z/OS
MQIA_TREE_LIFE_TIME	MQLONG	Czas życia nieużywanych tematów nieadministracyjnych	
MQIA_TRIGGER_INTERVAL	MQLONG	Interwał wyzwalacza	

IntAttrLiczba

Typ: MQLONG -wejście

Jest to liczba elementów w tablicy *IntAttrs*. Wartość zero jest poprawną wartością.

Jeśli parametr *IntAttrCount* jest co najmniej liczbą selektorów MQIA_* w parametrze *Selectors*, zwracane są wszystkie żądane atrybuty całkowitoliczbowe.

IntAttrs

Typ: MQLONG x *IntAttrCount* -wyjście

Jest to tablica wartości atrybutu całkowitoliczbowego *IntAttrCount*.

Wartości atrybutów całkowitych są zwracane w tej samej kolejności, w jakiej znajdują się selektory MQIA_* w parametrze *Selectors*. Jeśli tablica zawiera więcej elementów niż liczba selektorów MQIA_*, nadmiarowe elementy są niezmienione.

Jeśli *Hobj* reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, zwracana jest konkretna wartość MQIAV_NOT_APPLICABLE. Jest on zwracany dla odpowiedniego elementu w tablicy *IntAttrs*.

If the *IntAttrCount* or *SelectorCount* parameter is zero, *IntAttrs* is not referred to. W tym przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 assembler może mieć wartość NULL.

CharAttrDługość

Typ: MQLONG -wejście

Jest to długość w bajtach parametru *CharAttrs*.

CharAttrDługość musi być co najmniej równa sumie długości żądanych atrybutów znakowych (patrz [Selektory](#)). Wartość zero jest poprawną wartością.

CharAttrs

Typ: MQCHAR x *CharAttrLength* -wyjście

Jest to bufor, w którym zwracane są atrybuty znakowe, konkatelowane razem. Długość buforu jest nadawana przez parametr *CharAttrLength*.

Atrybuty znaków są zwracane w tej samej kolejności, co selektory MQCA_* w parametrze *Selectors*. Długość każdego łańcucha atrybutu jest stała dla każdego atrybutu (patrz sekcja [Selektory](#)), a wartość w niej jest dopełniona do prawej strony odstępami, jeśli jest to konieczne. Bufor może być większy niż wymagany, aby zawierał wszystkie żądane atrybuty znaków i dopełnianie. Liczba bajtów spoza ostatniej zwracanej wartości atrybutu nie została zmieniona.

Jeśli parametr *Hobj* reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, zwracany jest łańcuch znaków składający się w całości z gwiazdek (*). Gwiazdka jest zwracana jako wartość tego atrybutu w produkcie *CharAttrs*.

If the *CharAttrLength* or *SelectorCount* parameter is zero, *CharAttrs* is not referred to. W tym przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 assembler może mieć wartość NULL.

CompCode

Typ: MQLONG -wyjście

Kod zakończenia:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjscie

Jeśli *CompCode* to MQCC_OK:

MQRC_NONE

(0, X'000') Nie ma powodu do zgłaszania.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

(2008, X'7D8') Niewystarczająca ilość miejsca na atrybuty znaków.

MQRC_INT_ATTR_COUNT_TOO_SMALL

(2022, X'7E6') Niewystarczająca ilość miejsca na atrybuty całkowitoliczbowe.

MQRC_SELECTOR_NOT_FOR_TYPE

(2068, X'814') Selektor nie ma zastosowania do typu kolejki.

Jeśli *CompCode* to MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852') Nie można załadować modułu usługi adaptera.

MQRC_API_EXIT_ERROR

(2374, X'946') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_STRUC_FAILED

(2373, X'945') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') Długość atrybutów znakowych nie jest poprawna.

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') Łańcuch atrybutów znakowych nie jest poprawny.

MQRC_CICS_WAIT_FAILED

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_STOPPING

(2203, X'89B') Połączenie jest zamykane.

MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') Liczba atrybutów całkowitych nie jest poprawna.

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') Tablica atrybutów całkowitoliczbowych jest niepoprawna.

MQRC_NOT_OPEN_FOR_INQUIRE

(2038, X'7F6') Kolejka nie jest otwarta dla zapytania.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_DAMAGED

(2101, X'835') Obiekt jest uszkodzony.

MQRC_PAGESET_ERROR

(2193, X'891') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED

(2052, X'804') Kolejka została usunięta.

MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_STOPPING

(2162, X'872') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM

(2102, X'836') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SELECTOR_COUNT_ERROR

(2065, X'811') Liczba selektorów nie jest poprawna.

MQRC_SELECTOR_ERROR

(2067, X'813') Selektor atrybutu nie jest poprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED

(2066, X'812') Liczba zbyt dużych selektorów.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Zwracane wartości są obrazem stanu wybranych atrybutów. Nie ma gwarancji, że atrybuty pozostaną takie same, zanim aplikacja będzie mogła działać na zwrócone wartości.
2. Po otwarciu kolejki modelowej tworzona jest dynamiczna kolejka lokalna. Dynamiczna kolejka lokalna jest tworzona nawet wtedy, gdy kolejka modelowa zostanie otwarta w celu uzyskania informacji o jej atrybutach.

Atrybuty kolejki dynamicznej są w dużej mierze takie same, jak atrybuty kolejki modelowej w momencie tworzenia kolejki dynamicznej. Jeśli następnie zostanie użyte wywołanie MQINQ w tej kolejce, menedżer kolejek zwróci atrybuty kolejki dynamicznej, a nie atrybuty kolejki modelowej. Więcej informacji na temat atrybutów kolejki modelowej dziedziczonych przez kolejkę dynamiczną zawiera sekcja [Tabela 112 na stronie 848](#).
3. Jeśli sprawdzany obiekt jest kolejką aliasową, to wartości atrybutów zwracane przez wywołanie MQINQ są atrybutami kolejki aliasowej. Nie są to atrybuty kolejki podstawowej ani tematu, do którego alias jest rozstrzygany.
4. Jeśli sprawdzany obiekt jest kolejką klastra, atrybuty, które mogą być zapytania, zależą od sposobu otwierania kolejki:

- Istnieje możliwość otwarcia kolejki klastra dla zapytania oraz jednego lub większej liczby operacji wprowadzania, przeglądania lub ustawiania. Aby to zrobić, musi istnieć lokalna instancja kolejki klastra, aby możliwe było pomyślne wykonanie tej operacji. W tym przypadku atrybuty, które mogą zostać zapytane, są atrybutami, które są poprawne dla kolejek lokalnych.

Jeśli kolejka klastra jest otwarta dla zapytania bez wprowadzania, przeglądania lub ustawiania, wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068), jeśli podjęto próbę zapytania o atrybuty, które są poprawne tylko dla kolejek lokalnych, a nie dla kolejek klastra.

- Istnieje możliwość otwarcia kolejki klastra w celu uzyskania informacji podczas przekazywania podstawowej nazwy menedżera kolejek potężonego menedżera kolejek.

Aby to zrobić, musi istnieć lokalna instancja kolejki klastra, aby możliwe było pomyślne wykonanie tej operacji. Jeśli podstawowy menedżer kolejek nie zostanie przekazany, wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068), jeśli użytkownik podejmie próbę uzyskania informacji o atrybutach, które są poprawne tylko dla kolejek lokalnych, a nie kolejek klastra

- Jeśli kolejka klastra jest otwarta tylko do zapytania, a także do uzyskiwania informacji i danych wyjściowych, można określić tylko te atrybuty, które są wymienione na liście. W tym przypadku atrybut **QType** ma wartość MQQT_CLUSTER :

- MQCA_Q_DESC
- MQCA_Q_NAME
- MQIA_DEF_BIND
- MQIA_DEF_PERSISTENCE
- MQIA_DEF_PRIORITY
- MQIA_INHIBIT_PUT
- MQIA_Q_TYPE

Kolejkę klastra można otworzyć bez ustalonego powiązania. Można go otworzyć za pomocą programu MQ00_BIND_NOT_FIXED określonego w wywołaniu MQOPEN . Można również określić wartość MQ00_BIND_AS_Q_DEFi ustawić atrybut **DefBind** kolejki na wartość MQBND_BIND_NOT_FIXED. Jeśli kolejka klastra zostanie otwarta bez ustalonego powiązania, kolejne wywołania programu MQINQ dla kolejki mogą uzyskać dostęp do różnych instancji kolejki klastra. Jednak jest to typowe dla wszystkich instancji, które mają te same wartości atrybutów.

- Obiekt kolejki aliasowej może być zdefiniowany dla klastra. Ponieważ atrybuty TARGTYPE i TARGET nie są atrybutami klastra, proces przeprowadzający proces MQOPEN w kolejce aliasowej nie jest świadomy obiektu, do którego alias jest tłumaczona.

Podczas początkowego MQOPENkolejka aliasowa jest tłumaczona na menedżer kolejek i kolejkę w klastrze. Rozstrzyganie nazw odbywa się ponownie w zdalnym menedżerze kolejek i znajduje się w tym miejscu, że TARGTPYE kolejki aliasowej jest rozstrzygana.

Jeśli kolejka aliasowa jest tłumaczona na alias tematu, to publikowanie komunikatów umieszczonych w kolejce aliasowej odbywa się w tym zdalnym menedżerze kolejek.

Patrz sekcja Kolejki klastrów .

5. Można zapytać o liczbę atrybutów, a następnie ustawić niektóre z nich za pomocą wywołania MQSET . Aby program zapytywać i wydajnie ustawiać, należy ustawić atrybuty, które mają być ustawione na początku tablic selektorów. W takim przypadku te same tablice z licznymi zredukami mogą być używane w przypadku produktu MQSET.
6. Jeśli wystąpi więcej niż jedna z sytuacji ostrzegawczych (patrz parametr *CompCode*), zwrócony kod przyczyny jest pierwszym z nich na następującej liście, która ma zastosowanie:
 - a. MQRC_SELECTOR_NOT_FOR_TYPE
 - b. MQRC_INT_ATTR_COUNT_TOO_SMALL
 - c. MQRC_CHAR_ATTRS_TOO_SHORT

7. Poniższe informacje zawierają informacje na temat atrybutów obiektu:

- [“Atrybuty dla kolejek” na stronie 845](#)
- [“Atrybuty dla list nazw” na stronie 879](#)
- [“Atrybuty definicji procesów” na stronie 881](#)
- [“Atrybuty dla menedżera kolejek” na stronie 810](#)

Wywołanie C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,  
       CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */  
MQHOBJ   Hobj;            /* Object handle */  
MQLONG   SelectorCount;   /* Count of selectors */  
MQLONG   Selectors[n];    /* Array of attribute selectors */  
MQLONG   IntAttrCount;    /* Count of integer attributes */  
MQLONG   IntAttrs[n];     /* Array of integer attributes */  
MQLONG   CharAttrLength;  /* Length of character attributes buffer */  
MQCHAR   CharAttrs[n];    /* Character attributes */  
MQLONG   CompCode;        /* Completion code */  
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,  
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,  
                  CHARATTRS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Object handle  
01 HOBJ           PIC S9(9) BINARY.  
** Count of selectors  
01 SELECTORCOUNT PIC S9(9) BINARY.  
** Array of attribute selectors  
01 SELECTORS-TABLE.  
02 SELECTORS      PIC S9(9) BINARY OCCURS n TIMES.  
** Count of integer attributes  
01 INTATTRCOUNT PIC S9(9) BINARY.  
** Array of integer attributes  
01 INTATTRS-TABLE.  
02 INTATTRS       PIC S9(9) BINARY OCCURS n TIMES.  
** Length of character attributes buffer  
01 CHARATTRLENGTH PIC S9(9) BINARY.  
** Character attributes  
01 CHARATTRS      PIC X(n).  
** Completion code  
01 COMPCODE        PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON          PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,  
            IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl SelectorCount  fixed bin(31); /* Count of selectors */
dcl Selectors(n)   fixed bin(31); /* Array of attribute selectors */
dcl IntAttrCount   fixed bin(31); /* Count of integer attributes */
dcl IntAttrs(n)    fixed bin(31); /* Array of integer attributes */
dcl CharAttrLength fixed bin(31); /* Length of character attributes
                                buffer */
dcl CharAttrs      char(n);       /* Character attributes */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying
                                CompCode */

```

Wywołanie High Level Assembler

```

CALL MQINQ, (HCONN, HOBJ, SELECTORCOUNT, SELECTORS, INTATTRCOUNT, X
            INTATTRS, CHARATTRLENGTH, CHARATTRS, COMPCODE, REASON)

```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```

MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
CharAttrLength, CharAttrs, CompCode, Reason

```

Zadeklaruj parametry w następujący sposób:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim SelectorCount	As Long	'Count of selectors'
Dim Selectors	As Long	'Array of attribute selectors'
Dim IntAttrCount	As Long	'Count of integer attributes'
Dim IntAttrs	As Long	'Array of integer attributes'
Dim CharAttrLength	As Long	'Length of character attributes buffer'
Dim CharAttrs	As String	'Character attributes'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQINQMP-właściwość komunikatu Inquire

Wywołanie MQINQMP zwraca wartość właściwości komunikatu.

Składnia

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwyttem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy MQHC_UNASSOCIATED_HCONN, konieczne jest nawiązanie poprawnego połączenia w wątku, w którym znajduje się właściwość uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd MQRC_CONNECTION_BROKEN.

Hmsg

Typ: MQHMSG-wejście

Jest to uchwyt komunikatu, który ma zostać wyświetlony. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

InqPropOpts

Typ: MQIMPO-input/output

Szczegółowe informacje zawiera opis typu danych MQIMPO.

nazwa

Typ: MQCHARV-wejście/wyjście

Nazwa właściwości, która ma zostać zapytana.

Jeśli nie można znaleźć żadnej właściwości o tej nazwie, wywołanie nie powiedzie się, przyczyna: MQRC_PROPERTY_NOT_AVAILABLE.

Na końcu nazwy właściwości można użyć znaku wieloznacznego procentu (%). Znak wieloznaczny zastępuje zero lub więcej znaków, w tym znak kropki (.). Dzięki temu aplikacja może uzyskać dostęp do wartości wielu właściwości. Wywołaj komendę MQINQMP z opcją MQIMPO_INQ_FIRST, aby pobrać pierwszą zgodną właściwość i ponownie z opcją MQIMPO_INQ_NEXT, aby uzyskać następną pasującą właściwość. Jeśli nie są dostępne żadne dodatkowe właściwości, wywołanie nie powiedzie się i zostanie uruchomione wywołanie MQRC_PROPERTY_NOT_AVAILABLE. Jeśli pole *ReturnedName* struktury InqProp zostanie zainicjowane z adresem lub przesuniętą dla zwróconej nazwy właściwości, zostanie ona zakończona po powrocie z tabeli MQINQMP z nazwą właściwości, która została dopasowana. Jeśli pole *VSBufSize* w strukturze *ReturnedName* w strukturze InqPropOpts jest mniejsze niż długość zwróconej nazwy właściwości, to kod zakończenia jest ustawiony na wartość MQCC_FAILED z powodu MQRC_PROPERTY_NAME_TOO_BIG.

Właściwości, które mają znane synonimy, są zwracane w następujący sposób:

1. Właściwości z przedrostkiem "mqps." są zwracane jako nazwa właściwości IBM MQ. Na przykład "MQTopicString" jest nazwą zwracaną, a nie "mqps.Top".
2. Właściwości z przedrostkiem "jms." lub "mcd." są zwracane jako nazwa pola nagłówka JMS, na przykład: "JMSExpiration" to nazwa zwracana, a nie "jms.Exp".
3. Właściwości z przedrostkiem "usr." są zwracane bez tego przedrostka, na przykład zwracana jest wartość "Color", a nie "usr.Color".

Właściwości z synonimami są zwracane tylko jeden raz.

W języku programowania C następujące zmienne makra są zdefiniowane dla zapytania o wszystkie właściwości, a następnie wszystkie właściwości, które rozpoczynają się od "usr.":

MQPROP_INQUIRE_ALL

Sprawdź, czy wszystkie właściwości komunikatu są dostępne.

Wartość MQPROP_INQUIRE_ALL może być używana w następujący sposób:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR

Sprawdź wszystkie właściwości komunikatu, które zaczynają się od "usr.". Zwrócona nazwa jest zwracana bez użycia "usr." przedrostek.

Jeśli podana jest wartość MQIMP_INQ_NEXT, ale nazwa została zmieniona od czasu poprzedniego wywołania lub jest to pierwsze wywołanie, wówczas wartość MQIMPO_INQ_FIRST jest dorozumiana.

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja [Nazwy właściwości](#) i [Ograniczenia dotyczące nazw właściwości](#).

PropDesc

Typ: MQPD-wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym elementów, które są wykonywane, jeśli właściwość nie jest obsługiwana, kontekst komunikatu, do którego należy właściwość oraz jakie komunikaty należy skopiować do tej właściwości. Szczegółowe informacje na temat tej struktury zawiera sekcja [MQPD](#).

typ

Typ: MQLONG-input/output

W przypadku powrotu z wywołania MQINQMP ten parametr jest ustawiony na typ danych *Wartość*. Typ danych może mieć jedną z następujących wartości:

MQTYPE_BOOLEAN

Wartość boolowska.

MQTYPE_BYTE_STRING

łańcuch bajtowy.

MQTYPE_INT8

8-bitowa liczba całkowita ze znakiem.

MQTYPE_INT16

16-bitowa liczba całkowita ze znakiem.

MQTYPE_INT32

32-bitowa liczba całkowita ze znakiem.

MQTYPE_INT64

64-bitowa liczba całkowita ze znakiem.

MQTYPE_FLOAT32

32-bitowa liczba zmiennoprzecinkowa.

MQTYPE_FLOAT64

64-bitową liczbę zmiennopozycyjną.

MQTYPE_STRING

łańcuch znaków.

MQTYPE_NULL

Właściwość istnieje, ale ma wartość NULL.

Jeśli typ danych wartości właściwości nie zostanie rozpoznany, zwrócona zostanie wartość MQTYPE_STRING, a reprezentacja łańcuchowa wartości zostanie umieszczona w obszarze *Wartość*. Reprezentację łańcuchową typu danych można znaleźć w polu *TypeString* w parametrze *InqPropOpts*. Kod zakończenia ostrzeżenia jest zwracany z przyczyny MQRC_PROP_TYPE_NOT_SUPPORTED.

Dodatkowo, jeśli określono opcję MQIMPO_CONVERT_TYPE, wymagana jest konwersja wartości właściwości. Użyj opcji *Type* (Typ) jako danych wejściowych, aby określić typ danych, który ma być zwracany jako właściwość. Szczegółowe informacje na temat konwersji typów danych można znaleźć w opisie opcji [MQIMPO_CONVERT_TYPE](#) struktury [MQIMPO](#).

Jeśli nie zostanie wysłane żądanie konwersji typów, można użyć następującej wartości na wejściu:

MQTYPE_AS_SET

Wartość właściwości jest zwracana bez przekształcania jej typu danych.

ValueLength

Typ: MQLONG-wejście

Długość w bajtach obszaru *Wartość*. Podaj wartość zero dla właściwości, dla których nie jest wymagana zwracana wartość. Mogą to być właściwości, które zostały zaprojektowane przez aplikację

w celu posiadania wartości NULL lub pustego łańcucha. Należy również określić wartość zero, jeśli została określona opcja `MQIMPO_QUERY_LENGTH`. W tym przypadku nie jest zwracana żadna wartość.

wartość

Typ: `MQBYTEx ValueLength` -dane wyjściowe

Jest to obszar, który ma zawierać wartość właściwości `inquired`. Bufor powinien być wyrównany do granicy odpowiedniej dla zwracanej wartości. Niezastosowanie się do tej wartości może spowodować wystąpienie błędu, gdy wartość zostanie później uzyskana.

Jeśli wartość `ValueLength` jest mniejsza niż długość wartości właściwości, to jak większość wartości właściwości jest przenoszona do wartości `Value`, a wywołanie kończy się niepowodzeniem z kodem zakończenia `MQCC_FAILED` i przyczyna `MQRC_PROPERTY_VALUE_TOO_BIG`.

Zestaw znaków danych w polu *Wartość* jest nadawany przez pole `ReturnedCCSID` w parametrze `InqPropOpts`. Kodowanie danych w polu *Wartość* jest nadawane przez pole `ReturnedEncoding` w parametrze `InqPropOpts`.

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void; adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr `ValueLength` ma wartość zero, wartość `Value` nie jest przywołana, a jego wartość przekazywana przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

DataLength

Typ: `MQLONG`-wyjście

Jest to długość w bajtach rzeczywistej wartości właściwości, która została zwrócona w obszarze *Wartość*.

Jeśli wartość `DataLength` jest mniejsza niż długość wartości właściwości, wartość `DataLength` jest nadal wypełniona po powrocie z wywołania `MQINQMP`. Dzięki temu aplikacja może określić wielkość buforu wymaganego do uwzględnienia wartości właściwości, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Można również zwrócić następujące wartości.

Jeśli parametr `Type` jest ustawiony na wartość `MQTYPE_STRING` lub `MQTYPE_BYTE_STRING`:

MQVL_EMPTY_STRING

Właściwość istnieje, ale nie zawiera żadnych znaków ani bajtów.

CompCode

Typ: `MQLONG`-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: `MQLONG`-wyjście

Jeśli `CompCode` ma wartość `MQCC_OK`:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli `CompCode` to `MQCC_WARNING`:

MQRC_PROP_NAME_NOT_CONVERTED

(2492, X'09BC') Zwrócona nazwa właściwości nie została przekształcona.

MQRC_PROP_VALUE_NOT_CONVERTED

(2466, X'09A2') Wartość właściwości nie została przekształcona.

MQRC_PROP_TYPE_NOT_SUPPORTED

(2467, X'09A3') Typ danych właściwości nie jest obsługiwany.

Błąd formatu MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'086D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'07D4') Parametr Wartość nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parametr długości wartości nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') Parametr długości danych nie jest poprawny.

BŁĄD MQRC_IMPO_ERROR

(2464, X'09A0') Zapytanie o strukturę opcji właściwości komunikatu nie jest poprawne.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07F8') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PD_ERROR

(2482, X'09B2') Struktura deskryptora właściwości nie jest poprawna.

MQRC_PROP_CONV_NOT_SUPPORTED

(2470, X'09A6') Konwersja z rzeczywistego na żądany typ danych nie jest obsługiwana.

Błąd MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Niepoprawna nazwa właściwości.

MQRC_PROPERTY_NAME_TOO_BIG

(2465, X'09A1') Nazwa właściwości jest zbyt duża dla zwróconego buforu nazw.

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7) Właściwość nie jest dostępna.

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') Wartość właściwości jest zbyt duża dla obszaru Wartość.

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') Napotkano błąd formatu liczb w danych wartości.

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') Niepoprawny żądany typ właściwości.

MQRC_SOURCE_CCSID_ERROR, BŁĄD

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'0871 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;          /* Connection handle */
MQHMSG Hmsg;             /* Message handle */
MQIMPO InqPropOpts;      /* Options that control the action of MQINQMP */
MQCHARV Name;            /* Property name */
MQPD PropDesc;           /* Property descriptor */
MQLONG Type;             /* Property data type */
MQLONG ValueLength;      /* Length in bytes of the Value area */
MQBYTE Value[n];         /* Area to contain the property value */
MQLONG DataLength;       /* Length of the property value */
MQLONG CompCode;         /* Completion code */
MQLONG Reason;           /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Message handle
01 HMSG           PIC S9(18) BINARY.
** Options that control the action of MQINQMP
01 INQMSGOPTS.
   COPY CMQIMPOV.
** Property name
01 NAME.
   COPY CMQCHRVV.
** Property descriptor
01 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.
```

Wywołanie PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl InqPropOpts like MQIMPO; /* Options that control the action of MQINQMP */
dcl Name       like MQCHARV; /* Property name */
dcl PropDesc   like MQPD; /* Property descriptor */
dcl Type       fixed bin (31); /* Property data type */
dcl ValueLength fixed bin (31); /* Length in bytes of the Value area */
dcl Value      char (n); /* Area to contain the property value */
dcl DataLength fixed bin (31); /* Length of the property value */
dcl CompCode   fixed bin (31); /* Completion code */
dcl Reason     fixed bin (31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQINQMP, (HCONN, HMSG, INQMSGOPTS, NAME, PROPDSC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
INQMSGOPTS	CMQIMPOA	,	Options that control the action of MQINQMP
NAME	CMQCHRVA	,	Property name
PROPDSC	CMQPDA	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length in bytes of the VALUE area
VALUE	DS	CL(n)	Area to contain the property value
DATALENGTH	DS	F	Length of the property value
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQMHBUF-Przekształć uchwyt komunikatu w bufor

Wywołanie MQMHBUF przekształca uchwyt komunikatu w bufor i jest odwrotnym wywołaniem wywołania MQBUFMH.

Składnia

MQMHBUF (*Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer, DataLength, CompCode, Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwyttem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, w wątku usuwaniu uchwytu komunikatu musi zostać nawiązane poprawne połączenie. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie komendy MQRC_CONNECTION_BROKEN nie powiedzie się.

Hmsg

Typ: MQHMSG-wejście

Jest to uchwyt komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

MsgHBufOpts

Typ: MQMHBO-wejście

Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki bufor jest generowany z uchwytów komunikatów.

Szczegółowe informacje na ten temat zawiera sekcja [“MQMHBO-uchwyt komunikatu do opcji buforu”](#) na stronie 477.

nazwa

Typ: MQCHARV-wejście

Nazwa właściwości lub właściwości, które mają zostać umieszczone w buforze.

Jeśli nie można znaleźć żadnej właściwości zgodnej z nazwą, wywołanie nie powiedzie się i zostanie podana wartość MQRC_PROPERTY_NOT_AVAILABLE.

Aby umieścić w buforze więcej niż jedną właściwość, można użyć znaku wieloznacznego. W tym celu należy użyć znaku wieloznacznego "%" na końcu nazwy właściwości. Ten znak wieloznaczny jest zgodny z zero lub większą liczbę znaków, w tym znak '!' na końcu.

W języku programowania C następujące zmienne makra są zdefiniowane dla zapytania o wszystkie właściwości i wszystkie właściwości, które zaczynają się od 'usr':

MQPROP_INQUIRE_ALL

Umieść wszystkie właściwości komunikatu w buforze

MQPROP_INQUIRE_ALL_USR

Umieść wszystkie właściwości komunikatu, które rozpoczynają się od znaków usr. do buforu.

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja [Nazwy właściwości i Ograniczenia dotyczące nazw właściwości](#).

MsgDesc

Typ: MQMD-input/output

Struktura *MsgDesc* opisuje zawartość obszaru buforu.

W przypadku danych wyjściowych pola *Encoding*, *CodedCharSetId* i *Format* są ustawione tak, aby poprawnie opisywać kodowanie, identyfikator zestawu znaków i format danych w obszarze buforu, jak zostało to zapisane w wywołaniu.

Dane w tej strukturze znajdują się w zestawie znaków i kodowaniu aplikacji.

BufferLength

Typ: MQLONG-wejście

BufferLength to długość obszaru buforu (w bajtach).

Buforuj

Typ: MQBYTEExBufferDługość-wyjście

Bufor definiuje obszar, w którym mają być zawarte właściwości komunikatu. Bufor musi być wyrównany w 4-bajtowej granicy.

Jeśli wartość parametru *BufferLength* jest mniejsza niż długość wymagana do zapisania właściwości w produkcie *Bufor*, komenda MQMHBUF nie powiedzie się z wartością MQRC_PROPERTY_VALUE_TOO_BIG.

Zawartość buforu może się zmieniać nawet wtedy, gdy wywołanie nie powiedzie się.

DataLength

Typ: MQLONG-wyjście

DataLength to długość (w bajtach) zwracanych właściwości w buforze. Jeśli wartość jest równa zero, żadne właściwości nie są zgodne z wartością podaną w produkcie *Name* , a wywołanie nie powiedzie się, a kod przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

Jeśli wartość *BufferLength* jest mniejsza niż długość wymagana do zapisania właściwości w buforze, wywołanie MQMHBUF kończy się niepowodzeniem z właściwością MQRC_PROPERTY_VALUE_TOO_BIG, ale wartość ta jest nadal wprowadzana do produktu *DataLength*. Dzięki temu aplikacja może określić wielkość buforu wymaganego do dostosowania właściwości, a następnie ponownie wywołać wywołanie z wymaganym *BufferLength*.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_MHBO_ERROR

(2501, X'095C') Uchwyt komunikatu do struktury opcji buforu nie jest poprawny.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'07D4') Parametr buforu nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CONNECTION_BROKEN

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'07DA') Parametr długości danych nie jest poprawny.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

Błąd MQRC_MD_ERROR

(2026, X'07EA') deskryptor komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

Błąd MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Nazwa właściwości jest niepoprawna.

MQRC_PROPERTY_NOT_AVAILABLE

(2471, X'09A7') Właściwość nie jest dostępna.

MQRC_PROPERTY_VALUE_TOO_BIG

(2469, X'09A5') Wartość parametru BufferLength jest zbyt mała, aby można było zawierać określone właściwości.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;          /* Connection handle */
MQHMSG  Hmsg;            /* Message handle */
MQMHBO  MsgHBufOpts;    /* Options that control the action of MQMHBUF */
MQCHARV Name;           /* Property name */
MQMD     MsgDesc;        /* Message descriptor */
MQLONG   BufferLength;    /* Length in bytes of the Buffer area */
MQBYTE   Buffer[n];       /* Area to contain the properties */
MQLONG   DataLength;     /* Length of the properties */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Użycie notatek

MQMHBUF przekształca uchwyt komunikatu w bufor.

Można go używać z wyjściem interfejsu API MQGET w celu uzyskania dostępu do określonych właściwości, przy użyciu interfejsów API właściwości komunikatu, a następnie przekazać je z powrotem do aplikacji zaprojektowanej w celu użycia nagłówków MQRFH2, a nie uchwytów komunikatów.

To wywołanie jest odwrotnym wywołaniem wywołania MQBUFMH, którego można użyć do analizowania właściwości komunikatu z buforu do uchwytu komunikatu.

Wywołanie języka COBOL

```
CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                     BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Message handle
01 HMSG           PIC S9(18) BINARY.
** Options that control the action of MQMHBUF
01 MSGHBUFOPTS.
   COPY 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.

```

Wywołanie PL/I

```
call MQMHBUFF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hmsg       fixed bin(63); /* Message handle */
dcl MsgHBufOpts like MQMHBO; /* Options that control the action of MQMHBUFF */
dcl Name       like MQCHARV; /* Property name */
dcl MsgDesc    like MQMD;    /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer      char(n);      /* Area to contain the properties */
dcl DataLength fixed bin(31); /* Length of the properties */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

Wywołanie High Level Assembler

```
CALL MQMHBUFF, (HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
BUFFER,DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
MSGHBUFOPTS	CMQMHB		Options that control the action of MQMHBUFF
NAME	CMQCH		Property name
MSGDESC	CMQMD		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-obiekt otwarty

Wywołanie MQOPEN ustanawia dostęp do obiektu.

Poprawne są następujące typy obiektów:

- Kolejka (w tym listy dystrybucyjne)
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat

Składnia

MQOPEN (*Hconn*, *ObjDesc*, *Options*, *Hobj*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

ObjDesc

Typ: MQOD-input/output

Jest to struktura identyfikująca obiekt, który ma zostać otwarty. Szczegółowe informacje znajdują się w sekcji [“MQOD-deskryptor obiektu”](#) na stronie 479 .

Jeśli pole *ObjectName* w parametrze *ObjDesc* jest nazwą kolejki modelowej, dynamiczna kolejka lokalna jest tworzony z atrybutami kolejki modelowej; dzieje się tak niezależnie od opcji określonych w parametrze *Options* . Kolejne operacje przy użyciu *Hobj* zwróconego przez wywołanie MQOPEN są wykonywane w nowej kolejce dynamicznej, a nie w kolejce modelowej. Jest to prawda nawet w przypadku wywołań MQINQ i MQSET. Nazwa kolejki modelowej w parametrze *ObjDesc* jest zastępowana nazwą utworzonej kolejki dynamicznej. Typ kolejki dynamicznej jest określany na podstawie wartości atrybutu *DefinitionType* kolejki modelowej (patrz [“Atrybuty dla kolejek”](#) na stronie 845). Informacje na temat opcji zamykania, które mają zastosowanie do kolejek dynamicznych, zawiera opis wywołania MQCLOSE.

Opcje

Typ: MQLONG-wejście

Należy określić co najmniej jedną z następujących opcji:

- MQOO_BROWSE
- MQOO_INPUT_ * (tylko jedno z nich)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_ * (tylko jeden z nich)

Szczegółowe informacje na temat tych opcji można znaleźć w poniższej tabeli; inne opcje można określić w zależności od potrzeb. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe). Poprawne są kombinacje, które nie są poprawne. Wszystkie pozostałe kombinacje są poprawne. Dozwolone są tylko opcje, które mają zastosowanie do typu obiektu określonego przez *ObjDesc* . W poniższej tabeli przedstawiono poprawne opcje MQOPEN dla zapytań i tematów.

Opcja	Alias ¹	Lokalne i modelowe	Zdalny	Klaster inny niż lokalny	Lista dystrybucyjna	Temat
MQOO_INPUT_AS_Q_DEF	Tak	Tak	Nie	Nie	Nie	Nie

Opcja	Alias ¹	Lokalne i modelowe	Zdalny	Klaster inny niż lokalny	Lista dystrybucyjna	Temat
<u>MQOO_INPUT_SHARED</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_INPUT_EXCLUSIVE</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_OUTPUT</u>	Tak	Tak	Tak	Tak	Tak	Tak
<u>MQOO_BROWSE</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_CO_OP</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_INQUIRE</u>	Tak	Tak	<u>2</u>	Tak	Nie	Nie
<u>ZESTAW MQOO_SET</u>	Tak	Tak	<u>2</u>	Nie	Nie	Nie
<u>MQOO_BIND_ON_OPEN</u> ³	Tak	Tak	Tak	Tak	Tak	Nie
<u>MQOO_BIND_NOT_FIXED</u> ³	Tak	Tak	Tak	Tak	Tak	Nie
<u>MQOO_BIND_ON_GROUP</u> ³	Tak	Tak	Tak	Tak	Tak	Nie
<u>MQOO_BIND_AS_Q_DEF</u> ³	Tak	Tak	Tak	Tak	Tak	Nie
<u>MQOO_SAVE_ALL_CONTEXT</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_PASS_IDENTITY_CONTEXT</u>	Tak	Tak	Tak	Tak	Tak	<u>4</u>
<u>MQOO_PASS_ALL_CONTEXT</u>	Tak	Tak	Tak	Tak	Tak	Tak
<u>MQOO_SET_IDENTITY_CONTEXT</u>	Tak	Tak	Tak	Tak	Tak	<u>4</u>
<u>MQOO_SET_ALL_CONTEXT</u>	Tak	Tak	Tak	Tak	Tak	Tak
<u>MQOO_NO_READ_AHEAD</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_READ_AHEAD</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_READ_AHEAD_AS_Q_DEF</u>	Tak	Tak	Nie	Nie	Nie	Nie
<u>MQOO_ALTERNATE_USER_AUTHORITY</u>	Tak	Tak	Tak	Tak	Tak	Tak
<u>MQOO_FAIL_IF QUIESCING</u>	Tak	Tak	Tak	Tak	Tak	Tak
<u>MQOO_RESOLVE_LOCAL_Q</u>	Tak	Tak	Tak	Tak	Nie	Nie
<u>MQOO_RESOLVE_LOCAL_TOPIC</u>	Nie	Nie	Nie	Nie	Nie	Tak
<u>MQOO_NO_MULTICAST</u>	Nie	Nie	Nie	Nie	Nie	Tak

Uwagi:

1. Ważność opcji dla aliasów zależy od poprawności opcji kolejki, do której jest rozstrzygany alias.
2. Ta opcja jest poprawna tylko w przypadku lokalnej definicji kolejki zdalnej.
3. Ta opcja może być określona dla dowolnego typu kolejki, ale jest ignorowana, jeśli kolejka nie jest kolejką klastra. Jednak atrybut kolejki *DefBind* przestania kolejkę podstawową nawet wtedy, gdy kolejka aliasowa nie znajduje się w klastrze.
4. Atrybuty te mogą być używane z tematem, ale mają wpływ tylko na kontekst ustawiony dla zachowanego komunikatu, a nie na pola kontekstu wysyłane do dowolnego subskrybenta.

Opcje dostępu: Następujące opcje sterują typem operacji, które mogą być wykonywane na obiekcie:

MQOO_INPUT_AS_Q_DEF

Otwieranie kolejki w celu pobierania komunikatów za pomocą wartości domyślnej zdefiniowanej przez kolejkę.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Typ dostępu jest współużytkowany lub na wyłączność, w zależności od wartości atrybutu kolejki produktu *DefInputOpenOption*. Szczegółowe informacje zawiera sekcja “Atrybuty dla kolejek” na stronie 845.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INPUT_SHARED

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest aktualnie otwarta przez tę lub inną aplikację z opcją MQOO_INPUT_SHARED, ale kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta z opcją MQOO_INPUT_EXCLUSIVE.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INPUT_EXCLUSIVE

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie nie powiodło się z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla danych wejściowych dowolnego typu (MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_OUTPUT

Otwieranie kolejki w celu umieszczania komunikatów lub łańcucha tematu lub tematu w celu publikowania komunikatów.

Kolejka lub temat jest otwierany do użycia z kolejnymi wywołaniami MQPUT.

Wywołanie MQOPEN z tą opcją może się powieść, nawet jeśli atrybut kolejki produktu *InhibitPut* jest ustawiony na wartość MQQA_PUT_INHIBITED (choćby kolejne wywołania MQPUT nie powiodły się, gdy atrybut jest ustawiony na tę wartość).

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych i tematów.

Do tych opcji mają zastosowanie następujące uwagi:

- Można określić tylko jedną z tych opcji.
- Wywołanie MQOPEN z jedną z tych opcji może się powieść, nawet jeśli atrybut kolejki produktu *InhibitGet* jest ustawiony na wartość MQQA_GET_INHIBITED (choćby kolejne wywołania MQGET nie powiodły się, gdy atrybut jest ustawiony na tę wartość).
- Jeśli kolejka jest zdefiniowana jako niewspółużytkowalna (czyli atrybut kolejki *Shareability* ma wartość MQQA_NOT_SHAREABLE), próby otwarcia kolejki na potrzeby współużytkowanego dostępu są traktowane jako próby otwarcia kolejki z wyłącznym dostępem.
- Jeśli kolejka aliasowa jest otwierana przy użyciu jednej z tych opcji, test wyłącznego użycia (lub dla tego, czy inna aplikacja ma wyłączne użycie) jest dla kolejki podstawowej, do której alias jest tłumaczący.
- Te opcje nie są poprawne, jeśli *ObjectQMGrName* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w lokalnej definicji kolejki zdalnej używanej do aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_BROWSE

Otwórz kolejkę, aby przeglądać komunikaty.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET z jedną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Jest to dozwolone nawet wtedy, gdy kolejka jest obecnie otwarta dla MQOO_INPUT_EXCLUSIVE. Wywołanie MQOPEN z opcją MQOO_BROWSE tworzy kursor przeglądania i umieszcza je logicznie przed pierwszym komunikatem w kolejce; więcej informacji na ten temat zawiera sekcja MQGMO-pole opcji.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Nie jest on również poprawny, jeśli *ObjectQMGrName* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w lokalnej definicji kolejki zdalnej używanej do aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_CO_OP

Otwarty jako członek współpracujący z zestawem uchwytów.

Ta opcja jest poprawna tylko w przypadku opcji MQOO_BROWSE. Jeśli jest ona określona bez komendy MQOO_BROWSE, komenda MQOPEN zwraca wartość MQRC_OPTIONS_ERROR.

Zwracany uchwyt jest uważany za element współpracującego zestawu uchwytów dla kolejnych wywołań MQGET z jedną z następujących opcji:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INQUIRE

Otwórz obiekt, aby uzyskać dostęp do atrybutów.

Kolejka, lista nazw, definicja procesu lub menedżer kolejek są otwierane w celu użycia z kolejnymi wywołaniami MQINQ.

Ta opcja jest poprawna dla wszystkich typów obiektów innych niż listy dystrybucyjne. Wartość ta nie jest poprawna, jeśli *ObjectQMGrName* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_SET

Otwieranie kolejki w celu ustawienia atrybutów.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQSET.

Ta opcja jest poprawna dla wszystkich typów kolejek innych niż listy dystrybucyjne. Wartość ta nie jest poprawna, jeśli *ObjectQMGrName* jest nazwą lokalnej definicji kolejki zdalnej. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

Opcje powiązania: Następujące opcje mają zastosowanie, gdy otwierany obiekt jest kolejką klastra; te opcje sterują powiązaniem uchwytu kolejki z instancją kolejki klastra:

MQOO_BIND_ON_OPEN

Lokalny menedżer kolejek powiąże uchwyt kolejki z instancją kolejki docelowej po otwarciu kolejki. W wyniku tego wszystkie komunikaty umieszczone przy użyciu tego uchwytu są wysyłane do tej samej instancji kolejki docelowej, a także do tej samej trasy.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

MQOO_BIND_NOT_FIXED

Spowoduje to zatrzymanie menedżera kolejek lokalnych, który powiąże uchwyt kolejki z instancją kolejki docelowej. W wyniku tego kolejne wywołania MQPUT używające tego uchwytu wysyłają komunikaty do *różnych* instancji kolejki docelowej lub do tej samej instancji, ale na różne trasy. Umożliwia również zmianę instancji wybranej później przez lokalny menedżer kolejek, menedżer kolejek zdalnych lub agent kanału komunikatów (MCA), zgodnie z warunkami sieciowymi.

Uwaga: Aplikacje klienckie i aplikacje serwerowe, które muszą wymieniać *serię* komunikatów w celu zakończenia transakcji, nie mogą używać wartości MQOO_BIND_NOT_FIXED (lub MQOO_BIND_AS_Q_DEF, jeśli parametr *DefBind* ma wartość MQBND_BIND_NOT_FIXED), ponieważ kolejne komunikaty z serii mogą być wysyłane do różnych instancji aplikacji serwera.

Jeśli opcja MQOO_BROWSE lub jedna z opcji MQOO_INPUT_* jest określona dla kolejki klastra, menedżer kolejek jest zmuszony do wybrania lokalnej instancji kolejki klastra. Oznacza to, że powiązanie uchwytu kolejki jest stałe, nawet jeśli określono parametr MQOO_BIND_NOT_FIXED.

Jeśli wartość MQOO_INQUIRE została określona za pomocą komendy MQOO_BIND_NOT_FIXED, kolejne wywołania MQINQ korzystające z tego uchwytu mogą zapytać o różne instancje kolejki klastra, chociaż zwykle wszystkie instancje mają te same wartości atrybutów.

Wartość MQOO_BIND_NOT_FIXED jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

MQOO_BIND_ON_GROUP

Umożliwia aplikacji żądanie, aby grupa komunikatów była przydzielona do tej samej instancji docelowej.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

MQOO_BIND_AS_Q_DEF

Lokalny menedżer kolejek powiąże uchwyt kolejki w sposób zdefiniowany przez atrybut kolejki *DefBind*. Wartością tego atrybutu jest MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED lub MQBND_BIND_ON_GROUP.

Wartość MQOO_BIND_AS_Q_DEF jest wartością domyślną, jeśli nie określono wartości MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED lub MQOO_BIND_ON_GROUP.

Dokumentacja programu pomocy MQOO_BIND_AS_Q_DEF. Opcja ta nie jest przeznaczona dla żadnej z dwóch pozostałych opcji wiązania, ale ponieważ jej wartość jest równa zero, nie można jej wykryć.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

MQOO_SAVE_ALL_CONTEXT

Informacje o kontekście są powiązane z tym uchwytym kolejki. Te informacje są ustawiane na podstawie kontekstu dowolnego komunikatu pobranego przy użyciu tego uchwytu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Te informacje o kontekście mogą być przekazywane do komunikatu umieszczanego w kolejce przy użyciu wywołań MQPUT lub MQPUT1. Zapoznaj się z opcjami MQPMO_PASS_IDENTITY_CONTEXT i MQPMO_PASS_ALL_CONTEXT opisanymi w sekcji [“MQPMO-Put-message, opcje”](#) na stronie 500.

Dopóki komunikat nie zostanie pomyślnie pobrany, nie można przekazać kontekstu do komunikatu umieszczanego w kolejce.

Komunikat pobrany przy użyciu jednej z opcji przeglądania MQGMO_BROWSE_* nie ma zapisanych informacji o kontekście (choć pola kontekstu w parametrze *MsgDesc* są ustawiane po przeglądowaniu).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Należy podać jedną z opcji MQOO_INPUT_*.

MQOO_PASS_IDENTITY_CONTEXT

Umożliwia to określenie opcji MQPMO_PASS_IDENTITY_CONTEXT w parametrze *PutMsgOpts*, gdy komunikat jest umieszczany w kolejce. Pozwala to na przesłanie informacji o kontekście tożsamości z kolejki wejściowej, która została otwarta za pomocą opcji MQOO_SAVE_ALL_CONTEXT. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

MQOO_PASS_ALL_CONTEXT

Umożliwia to określenie opcji MQPMO_PASS_ALL_CONTEXT w parametrze *PutMsgOpts* , gdy komunikat jest umieszczany w kolejce. Pozwala to na przesłanie informacji o kontekście tożsamości i pochodzenia z kolejki wejściowej, która została otwarta za pomocą opcji MQOO_SAVE_ALL_CONTEXT. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Ta opcja implikuje wartość MQOO_PASS_IDENTITY_CONTEXT, która nie musi być określona. Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

MQOO_SET_IDENTITY_CONTEXT,

Umożliwia to określenie opcji MQPMO_SET_IDENTITY_CONTEXT w parametrze *PutMsgOpts* , gdy komunikat jest umieszczany w kolejce. To daje komunikat do informacji o kontekście tożsamości zawartych w parametrze *MsgDesc* określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Ta opcja implikuje wartość MQOO_PASS_IDENTITY_CONTEXT, która nie musi być określona. Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

MQOO_SET_ALL_CONTEXT

Umożliwia to określenie opcji MQPMO_SET_ALL_CONTEXT w parametrze *PutMsgOpts* , gdy komunikat jest umieszczany w kolejce. To daje komunikat informacje o tożsamości i kontekście pochodzenia zawarte w parametrze *MsgDesc* określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Ta opcja oznacza następujące opcje, które nie muszą być określone:

- MQOO_PASS_IDENTITY_CONTEXT,
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT,

Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

Opcje odczytu z wyprzedzeniem:

W przypadku wywołania MQOPEN z opcją MQOO_READ_AHEAD klient IBM MQ umożliwia odczyt z wyprzedzeniem, jeśli spełnione są pewne warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi zostać skompilowana i powiązana z wątkowymi bibliotekami klienta MQI IBM MQ.
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Poniższe opcje kontrolują, czy komunikaty nietrwałe są wysyłane do klienta przed ich żądaniem. Do opcji odczytu z wyprzedzeniem mają zastosowanie następujące uwagi:

- Można określić tylko jedną z tych opcji.
- Te opcje są poprawne tylko dla kolejek lokalnych, aliasowych i modelowych. Nie są one poprawne w przypadku kolejek zdalnych, list dystrybucyjnych, tematów lub menedżerów kolejek.

- Te opcje mają zastosowanie tylko wtedy, gdy określono również jedną z opcji MQOO_BROWSE, MQOO_INPUT_SHARED i MQOO_INPUT_EXCLUSIVE, chociaż nie jest to błąd w celu określenia tych opcji z parametrem MQOO_INQUIRE lub MQOO_SET.
- Jeśli aplikacja nie jest uruchomiona jako klient IBM MQ, opcje te są ignorowane.

MQOO_NO_READ_AHEAD

Komunikaty nietrwałe nie są wysyłane do klienta, zanim aplikacja je zażąda.

MQOO_READ_AHEAD

Komunikaty nietrwałe są wysyłane do klienta, zanim aplikacja je zażąda.

MQOO_READ_AHEAD_AHEAD_AS_Q_DEF

Zachowanie odczytu z wyprzedzeniem jest określane przez domyślny atrybut odczytu z wyprzedzeniem dla otwieranej kolejki. Jest to wartość domyślna.

Inne opcje: Następujące opcje kontrolują sprawdzanie autoryzacji, co się dzieje, gdy menedżer kolejek jest wygaszany, niezależnie od tego, czy ma być rozstrzygana nazwa kolejki lokalnej, czy rozsyłanie grupowe:

MQOO_ALTERNATE_USER_AUTHORITY

Pole *AlternateUserId* w parametrze *ObjDesc* zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności wywołania MQOPEN. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy *AlternateUserId* jest autoryzowany do otwarcia obiektu przy użyciu określonych opcji dostępu, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma do tego uprawnienia. Nie dotyczy to jednak żadnych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym aplikacja jest uruchomiona.

Ta opcja jest poprawna dla wszystkich typów obiektów.

MQOO_FAIL_IF QUIESCING

Wywołanie MQOPEN nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania.

W systemie z/OS dla aplikacji CICS lub IMS ta opcja również wymusza, że wywołanie MQOPEN nie powiedzie się, jeśli połączenie jest w stanie wygaszania.

Ta opcja jest poprawna dla wszystkich typów obiektów.

Więcej informacji na temat kanałów klienta zawiera sekcja [Przegląd produktu IBM MQ MQI clients](#).

MQOO_RESOLVE_LOCAL_Q

Wypełnij wartość ResolvedQName w strukturze MQOD, podając nazwę kolejki lokalnej, która została otwarta. Podobnie nazwa obiektu ResolvedQMgr jest wypełniona nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną. Jeśli struktura MQOD jest mniejsza niż wersja 3, parametr MQOO_RESOLVE_LOCAL_Q jest ignorowany, gdy nie jest zwracany żaden błąd.

Kolejka lokalna jest zawsze zwracana, gdy otwarta jest kolejka lokalna, alias lub kolejka modelowa, ale nie jest to sytuacja, gdy na przykład kolejka zdalna lub nielokalna kolejka klastra jest otwierana bez opcji MQOO_RESOLVE_LOCAL_Q. Nazwa ResolvedQName i ResolvedQMgr jest wypełniona nazwą RemoteQName i RemoteQMgr nazwą znalezionej w definicji kolejki zdalnej lub podobnie z wybraną zdalną kolejką klastra.

Jeśli podczas otwierania zostanie podana wartość MQOO_RESOLVE_LOCAL_Q, na przykład kolejka zdalna, ResolvedQName jest kolejką transmisji, do której umieszczane są komunikaty. Nazwa obiektu ResolvedQMgr jest wypełniona nazwą lokalnego menedżera kolejek udostępniającego kolejkę transmisji.

Jeśli użytkownik ma uprawnienia do przeglądania, wprowadzania danych lub danych wyjściowych w kolejce, ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQOPEN. Nie jest wymagane żadne uprawnienie specjalne.

Ta opcja jest poprawna tylko dla kolejek i menedżerów kolejek.

MQOO_RESOLVE_LOCAL_TOPIC

Wypełnij pole ResolvedQName w strukturze MQOD, podając nazwę tematu administracyjnego, który został otwarty.

MQOO_NO_MULTICAST

Komunikaty publikacji nie są wysyłane przy użyciu rozsyłania grupowego.

Ta opcja jest poprawna tylko w przypadku opcji MQOO_OUTPUT. Jeśli jest ona określona bez komendy MQOO_OUTPUT, komenda MQOPEN zwraca wartość MQRC_OPTIONS_ERROR.

Ta opcja jest poprawna tylko dla tematu.

Hobj

Typ: MQHOBJ-wyjście

Ten uchwyt reprezentuje dostęp, który został utworzony dla obiektu. Musi być ona określona przy kolejnych wywołaniach programu IBM MQ , które działają na obiekcie. Traci ona ważność po wywołaniu wywołania MQCLOSE lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, kończy działanie.

Zasięg zwróconego uchwytu obiektu jest taki sam, jak zasięg uchwytu połączenia określonego w wywołaniu. Informacje na temat zasięgu uchwytu można znaleźć w sekcji [Parametr MQCONN-Hconn](#) .

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ALIAS_BASE_Q_TYPE_ERROR

(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_NOT_AVAILABLE

(2345, X' 929 ') Obiekt sprzęgający nie jest dostępny.

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') Sprawdzanie autoryzacji struktury CF (Coupling-Facility) nie powiodło się.

MQRC_CF_STRUC_ERROR

(2349, X'92D') Struktura CF (Coupling-Facility) jest niepoprawna.

MQRC_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X' 945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') Lista struktury narzędzia CF-nagłówków w użyciu.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

MQRC_CLUSTER_PUT_INHIBITED

(2268, X'8DC') Wywołania umieszczenia zablokowano dla wszystkich kolejek w klastrze.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') Błąd zasobu klastra.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE

(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

MQRC_DEF_XMIT_Q_USAGE_ERROR

(2199, X'897 ') Domyślny błąd wykorzystania kolejki transmisji.

Błąd MQRC_DYNAMIC_Q_NAME_ERROR

(2011, X'7DB') Nazwa kolejki dynamicznej nie jest poprawna.

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'7E1') Nie ma więcej dostępnych uchwytów.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

MQRC_NAME_IN_USE

(2201, X'899 ') Nazwa w użyciu.

MQRC_NAME_NOT_VALID_FOR_TYPE

(2194, X'892 ') Nazwa obiektu nie jest poprawna dla typu obiektu.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_OBJECT_ALREADY_EXISTS

(2100, X'834 ') Obiekt istnieje.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

MQRC_OBJECT_IN_USE

(2042, X'7FA') Obiekt jest już otwarty z opcjami powodujących konflikt.

MQRC_OBJECT_LEVEL_NIEZGODNY

(2360, X' 938 ') Poziom obiektu nie jest kompatybilny.

MQRC_OBJECT_NAME_ERROR

(2152, X'868 ') Nazwa obiektu nie jest poprawna.

MQRC_OBJECT_NOT_UNIQUE

(2343, X' 927 ') Obiekt nie jest unikalny.

Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR

(2153, X'869 ') Nazwa menedżera kolejek obiektu nie jest poprawna.

MQRC_OBJECT_RECORDS_ERROR

(2155, X'86B') Rekordy obiektów nie są poprawne.

MQRC_OBJECT_STRING_ERROR,

(2441, X'0989 ') Pole Objectstring nie jest poprawne

MQRC_OBJECT_TYPE_ERROR

(2043, X'7FB') Typ obiektu nie jest poprawny.

BŁĄD MQRC_OD_ERROR

(2044, X'7FC') Struktura deskryptora obiektu nie jest poprawna.

MQRC_OPTION_NOT_VALID_FOR_TYPE

(2045, X'7FD') Opcja nie jest poprawna dla typu obiektu.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_TYPE_ERROR

(2057, X'809 ') Typ kolejki nie jest poprawny.

BŁĄD MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Liczba obecnie niepoprawnych rekordów.

MQRC_REMOTE_Q_NAME_ERROR

(2184, X'888 ') Nazwa zdalnej kolejki nie jest poprawna.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_RESPONSE_RECORDS_ERROR,

(2156, X'86C') Rekordy odpowiedzi nie są poprawne.

MQRC_SECURITY_ERROR,

(2063, X'80F') Wystąpił błąd zabezpieczeń.

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') Wywołanie MQOPEN, MQPUT1 lub MQSUB zostało wydane, ale podano łańcuch wyboru, który zawierał błąd składniowy.

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UNKNOWN_ALIAS_BASE_Q

(2082, X'822 ') Nieznana kolejka podstawowa aliasu.

MQRC_UNKNOWN_DEF_XMIT_Q

(2197, X'895 ') Nieznana domyślna kolejka transmisji.

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825 ') Nieznana nazwa obiektu.

MQRC_UNKNOWN_OBJECT_Q_MGR

(2086, X'826 ') Nieznany menedżer kolejek obiektów.

MQRC_UNKNOWN_REMOTE_Q_MGR

(2087, X'827 ') Nieznany zdalny menedżer kolejek.

MQRC_UNKNOWN_XMIT_Q

(2196, X'894 ') Nieznana kolejka transmisji.

MQRC_WRONG_CF_LEVEL

(2366, X'93E') Struktura CF (Coupling-Facility) jest poziomem błędnym.

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') Kolejka transmisji nie jest lokalna.

MQRC_XMIT_Q_USAGE_ERROR,

(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

Szczegółowe informacje na temat tych kodów można znaleźć w:

-  [Komunikaty, zakończenie i kody przyczyny produktu IBM MQ for z/OS dla produktu IBM MQ for z/OS.](#)
- [Komunikaty i kody przyczyny dla wszystkich innych platform IBM MQ z wyjątkiem z/OS.](#)

Ogólne uwagi dotyczące użycia

1. Otwarty obiekt jest jednym z następujących:

- Kolejka do:
 - Pobieranie lub przeglądanie komunikatów (za pomocą wywołania MQGET)
 - Umieszczanie komunikatów (za pomocą wywołania MQPUT)
 - Sprawdź atrybuty kolejki (za pomocą wywołania MQINQ)
 - Ustaw atrybuty kolejki (za pomocą wywołania MQSET)

Jeśli kolejka nazwana jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Patrz parametr *ObjDesc* opisany w sekcji “MQOPEN-obiekt otwarty” na stronie 742.

Lista dystrybucyjna jest specjalnym typem obiektu kolejki, który zawiera listę kolejek. Można je otwierać w celu umieszczania komunikatów, ale nie do pobierania ani przeglądania komunikatów, a także do uzyskiwania informacji lub ustawiania atrybutów. Więcej informacji na ten temat zawiera uwaga o składni 8.

Kolejka, która ma QSGDISP (GROUP) , jest specjalnym typem definicji kolejki, którego nie można używać z wywołaniami MQOPEN lub MQPUT1 .

- Lista nazw do zapytania o nazwy kolejek na liście (za pomocą wywołania MQINQ).
 - Definicja procesu do zapytania o atrybuty procesu (za pomocą wywołania MQINQ).
 - Menedżer kolejek, który ma uzyskać informacje na temat atrybutów menedżera kolejek lokalnych (za pomocą wywołania MQINQ).
 - Temat do publikowania komunikatu (za pomocą wywołania MQPUT)
2. Aplikacja może otworzyć ten sam obiekt więcej niż jeden raz. Dla każdego otwartego uchwytu zwracany jest inny uchwyt obiektu. Każdy zwracany uchwyt może być użyty dla funkcji, dla których wykonano odpowiednie otwarcie.
 3. Jeśli otwierany obiekt jest kolejką inną niż kolejka klastra, wówczas wszystkie rozstrzygnięcie nazw w lokalnym menedżerze kolejek odbywa się w czasie wywołania MQOPEN. Może to obejmować:
 - Rozstrzygnięcie nazwy lokalnej definicji kolejki zdalnej na nazwę menedżera kolejek zdalnych oraz nazwę, pod którą ta kolejka jest znana w zdalnym menedżerze kolejek
 - Rozstrzygnięcie nazwy zdalnego menedżera kolejek na nazwę lokalnej kolejki transmisji
 - (tylko w wersji z/OS) Rozdzielczość zdalnej nazwy menedżera kolejek na nazwę współużytkowanej kolejki transmisji używanej przez agenta IGQ (dotyczy tylko tego, czy lokalne i zdalne menedżery kolejek należą do tej samej grupy współużytkowania kolejki)
 - Rozdzielczość aliasu do nazwy kolejki podstawowej lub obiektu tematu.

Należy jednak pamiętać, że kolejne wywołania MQINQ lub MQSET dla uchwytu odnoszą się wyłącznie do nazwy, która została otwarta, a nie do obiektu, który ma miejsce po rozstrzygnięcie nazwy. Na przykład, jeśli otwarto obiekt jest aliasem, atrybuty zwracane przez wywołanie MQINQ są atrybutami aliasu, a nie atrybutami kolejki podstawowej lub obiektu tematu, do którego alias jest tłumaczący.

Jeśli otwierany obiekt jest kolejką klastra, rozstrzygnięcie nazwy może nastąpić w momencie wywołania MQOPEN lub zostać odroczone do czasu późniejszego. Punkt, w którym występuje rozstrzygnięcie, jest kontrolowany przez opcje MQOO_BIND_ * określone w wywołaniu MQOPEN:

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED
- MQOO_BIND_AS_Q_DEF
- MQOO_BIND_ON_GROUP

Więcej informacji na temat rozstrzygnięcia nazw kolejek klastra zawiera sekcja Rozdzielczość nazw .

4. Wywołanie MQOPEN z opcją MQOO_BROWSE tworzy kursor przeglądania, który jest używany z wywołaniami MQGET, które określają uchwyt obiektu i jedną z opcji przeglądania. Pozwala to na skanowanie kolejki bez zmiany jej zawartości. Komunikat, który został znaleziony podczas przeglądania, może zostać usunięty z kolejki za pomocą opcji MQGMO_MSG_UNDER_CURSOR.

Wiele kursorów przeglądania może być aktywnych dla pojedynczej aplikacji, wydając kilka żądań MQOPEN dla tej samej kolejki.

5. Aplikacje uruchomione przez monitor wyzwalacza są przekazywane do nazwy kolejki powiązanej z aplikacją, gdy aplikacja jest uruchomiona. Tę nazwę kolejki można określić w parametrze *ObjDesc*, aby otworzyć kolejkę. Więcej informacji na ten temat zawiera sekcja [“MQTMC2 -komunikat wyzwalacza 2 \(format znakowy\)”](#) na stronie 610.
6. W systemie IBM aplikacje działające w trybie zgodności są automatycznie połączone z menedżerem kolejek przy użyciu pierwszego wywołania MQOPEN wydanego przez aplikację (jeśli aplikacja nie nawiązała jeszcze połączenia z menedżerem kolejek przy użyciu wywołania MQCONN).

Aplikacje, które nie działają w trybie zgodności, muszą wywołać wywołanie MQCONN lub MQCONNX w celu jawnego nawiązania połączenia z menedżerem kolejek przed użyciem wywołania MQOPEN w celu otwarcia obiektu.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Opcje odczytu z wyprzedzeniem

W przypadku wywołania MQOPEN z opcją MQOO_READ_AHEAD klient IBM MQ umożliwia odczyt z wyprzedzeniem, jeśli spełnione są pewne warunki. Są one następujące:

- Zarówno klient, jak i menedżer kolejek zdalnych muszą być w wersji WebSphere MQ 7 lub nowszej.
- Aplikacja kliencka musi zostać skompilowana i powiązana z wątkowymi bibliotekami klienta MQI IBM MQ.
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Poniższe uwagi mają zastosowanie do korzystania z opcji odczytu z wyprzedzeniem.

1. Opcje odczytu z wyprzedzeniem mają zastosowanie tylko wtedy, gdy określono jedną i tylko jedną z opcji MQOO_BROWSE, MQOO_INPUT_SHARED i MQOO_INPUT_EXCLUSIVE. Błąd nie jest zgłaszany, jeśli opcje odczytu z wyprzedzeniem są określone za pomocą opcji MQOO_INQUIRE lub MQOO_SET.
2. Funkcja odczytu z wyprzedzeniem nie jest włączona w przypadku żądania, jeśli opcje używane w przypadku pierwszego wywołania MQGET nie są obsługiwane w celu użycia z wyprzedzeniem odczytu. Ponadto funkcja odczytu z wyprzedzeniem jest wyłączona, gdy klient łączy się z menedżerem kolejek, który nie obsługuje odczytu z wyprzedzeniem.
3. Jeśli aplikacja nie jest uruchomiona jako klient IBM MQ, opcje odczytu z wyprzedzeniem są ignorowane.

Kolejki klastra

Poniższe uwagi dotyczą korzystania z kolejek klastra.

1. Gdy kolejka klastra jest otwierana po raz pierwszy, a lokalny menedżer kolejek nie jest pełnym menedżerem kolejek repozytorium, lokalny menedżer kolejek uzyskuje informacje na temat kolejki klastra z pełnego menedżera kolejek repozytorium. Gdy sieć jest zajęta, może upłynąć kilka sekund, aby lokalny menedżer kolejek otrzymał wymagane informacje z menedżera kolejek repozytorium. W wyniku tego aplikacja wywołująca wywołanie MQOPEN może mieć do 10 sekund oczekiwania przed zwróceniem sterowania z wywołania MQOPEN. Jeśli lokalny menedżer kolejek nie otrzyma w tym czasie wymaganych informacji na temat kolejki klastra, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_CLUSTER_RESOLUTION_ERROR.
2. Gdy kolejka klastra jest otwarta i istnieje wiele instancji kolejki w klastrze, otwarcie instancji zależy od opcji określonych w wywołaniu MQOPEN:
 - Jeśli podane opcje zawierają dowolną z następujących opcji:

- MQOO_BROWSE
- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED
- MQOO_SET

Otwarto instancję kolejki klastra, która musi być instancją lokalną. Jeśli nie istnieje lokalna instancja kolejki, wywołanie MQOPEN nie powiedzie się.

- Jeśli podane opcje nie zawierają żadnej z opisanych wcześniej opcji, należy uwzględnić jedną lub obie z poniższych opcji:
 - MQOO_INQUIRE
 - MQOO_OUTPUT

Instancja została otwarta, jeśli istnieje jedna instancja, a w przeciwnym razie instancja zdalna (jeśli używana jest wartość domyślna CLWLUSEQ). Instancja wybrana przez menedżer kolejek może jednak zostać zmieniona przez wyjście obciążenia klastra (jeśli istnieje).

3. Jeśli istnieje subskrypcja kolejki, ale nie została ona potwierdzona przez pełne repozytorium, obiekt nie znajduje się w klastrze, a wywołanie nie powiedzie się. Kod przyczyny to MQRC_OBJECT_NAME.

Więcej informacji na temat kolejek klastra zawiera sekcja [Kolejki klastrów](#).

Lista dystrybucyjna

Do korzystania z list dystrybucyjnych stosuje się następujące uwagi.

Listy dystrybucyjne są obsługiwane w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączone z tymi systemami.

1. Pola w strukturze MQOD muszą być ustawione w następujący sposób podczas otwierania listy dystrybucyjnej:
 - *Version* musi mieć wartość MQOD_VERSION_2 lub większą.
 - *ObjectType* musi mieć wartość MQOT_Q.
 - Wartość *ObjectName* musi być pusta lub zawierać łańcuch pusty.
 - Wartość *ObjectQMGrName* musi być pusta lub zawierać łańcuch pusty.
 - Wartość *RecsPresent* musi być większa od zera.
 - Jeden z elementów *ObjectRecOffset* i *ObjectRecPtr* musi być zerem, a drugi niezerem.
 - Nie więcej niż jeden z serwerów *ResponseRecOffset* i *ResponseRecPtr* może być niezerowy.
 - Muszą istnieć rekordy obiektów *RecsPresent*, które są adresowane zarówno przez produkt *ObjectRecOffset*, jak i *ObjectRecPtr*. Rekordy obiektów muszą być ustawione na nazwy kolejek docelowych, które mają zostać otwarte.
 - Jeśli jeden z elementów *ResponseRecOffset* i *ResponseRecPtr* jest niezerowy, muszą istnieć rekordy odpowiedzi *RecsPresent*. Są one ustawiane przez menedżer kolejek, jeśli wywołanie zostało zakończone z kodem przyczyny MQRC_MULTIPLE_UZASADNIENIE.

Do otwarcia pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej, można użyć komendy version-2 MQOD, upewniając się, że parametr *RecsPresent* ma wartość zero.

2. W parametrze *Options* poprawne są tylko następujące opcje otwierania:
 - MQOO_OUTPUT
 - MQOO_PASS_*_KONTEKST
 - MQOO_SET_*_CONTEXT
 - MQOO_ALTERNATE_USER_AUTHORITY
 - MQOO_FAIL_IF QUIESCING

3. Kolejkami docelowymi na liście dystrybucyjnej mogą być kolejki lokalne, aliasy lub kolejki zdalne, ale nie mogą być kolejkami modelowymi. Jeśli określona jest kolejka modelowa, kolejka ta nie zostanie otwarta, a kod przyczyny MQRC_Q_TYPE_ERROR. Nie powoduje to jednak, że inne kolejki na liście zostaną otwarte pomyślnie.
4. Kod zakończenia i parametry kodu przyczyny są ustawione w następujący sposób:
 - Jeśli operacje otwarcia dla kolejek na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, parametry kodu zakończenia i kodu przyczyny zostaną ustawione tak, aby opisywać wspólny wynik. W tym przypadku nie są ustawione rekordy odpowiedzi MQRR (jeśli aplikacja jest udostępniana przez aplikację).

Na przykład, jeśli każde otwarcie powiedzie się, kod zakończenia jest ustawiony na wartość MQCC_OK, a kod przyczyny jest ustawiony na MQRC_NONE;; jeśli każde otwarcie się nie powiedzie, ponieważ żadna z tych kolejek nie istnieje, parametry są ustawiane na MQCC_FAILED i MQRC_UNKNOWN_OBJECT_NAME.
 - Jeśli otwarte operacje dla kolejek na liście dystrybucyjnej nie wszystkie powiodą się lub nie powiodą się w ten sam sposób:
 - Parametr kodu zakończenia jest ustawiony na wartość MQCC_WARNING, jeśli co najmniej jedno otwarcie powiodło się, a dla parametru MQCC_FAILED, jeśli wszystkie nie powiodły się.
 - Parametr kodu przyczyny jest ustawiony na wartość MQRC_MULTIPLE_UZASADNIENIE.
 - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.
5. Po pomyślnym otwarciu listy dystrybucyjnej uchwyt *Hobj* zwracany przez wywołanie może być użyty w kolejnych wywołaniach MQPUT w celu umieszczenia komunikatów w kolejkach na liście dystrybucyjnej, a w wywołaniu MQCLOSE w celu uzyskania dostępu do listy dystrybucyjnej. Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest MQCO_NONE.

Wywołanie MQPUT1 może również zostać użyte do umieszczenia komunikatu na liście dystrybucyjnej; struktura MQOD definiująca kolejki na liście jest określona jako parametr w wywołaniu.
6. Każde pomyślne otwarcie miejsca docelowego na liście dystrybucyjnej jest liczone jako osobny uchwyt podczas sprawdzania, czy aplikacja przekroczyła dozwoloną maksymalną liczbę uchwytów (patrz atrybut menedżera kolejek *MaxHandles*). Jest to prawda, nawet jeśli dwa lub więcej miejsc docelowych na liście dystrybucyjnej jest rozstrzygane do tej samej kolejki fizycznej. Jeśli wywołanie MQOPEN lub MQPUT1 dla listy dystrybucyjnej spowodowałoby, że liczba uchwytów używanych przez aplikację przekroczy *MaxHandles*, wywołanie nie powiedzie się i zostanie użyty kod przyczyny MQRC_HANDLE_NOT_AVAILABLE.
7. Każdy otwarty cel, który został pomyślnie otwarty, ma wartość atrybutu *OpenOutputCount* zwiększoną o jeden. Jeśli co najmniej dwa miejsca docelowe znajdujące się na liście dystrybucyjnej są rozstrzygane do tej samej kolejki fizycznej, atrybut *OpenOutputCount* jest zwiększany o liczbę miejsc docelowych znajdujących się na liście dystrybucyjnej, które są rozstrzygane do tej kolejki.
8. Każda zmiana w definicjach kolejek, które spowodowałyby, że uchwyt stał się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana ścieżki rozdzielczej), nie powoduje, że uchwyt listy dystrybucyjnej staje się niepoprawny. Jednak powoduje to niepowodzenie tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnych wywołaniach wywołania MQPUT.
9. Lista dystrybucyjna może zawierać tylko jedno miejsce docelowe.

Kolejki zdalne

Do korzystania z kolejek zdalnych mają zastosowanie następujące uwagi.

Kolejka zdalna może być określona na jeden z dwóch sposobów w parametrze *ObjDesc* tego wywołania.

- Określając parametr *ObjectName*, nazwę lokalnej definicji kolejki zdalnej. W tym przypadku program *ObjectQMgrName* odwołuje się do lokalnego menedżera kolejek i może zostać określony jako pusty lub (w języku programowania C) łańcuch pusty.

Sprawdzenie poprawności zabezpieczeń wykonywane przez lokalny menedżer kolejek sprawdza, czy użytkownik jest uprawniony do otwarcia lokalnej definicji kolejki zdalnej.

- Określając parametr *ObjectName* , nazwę kolejki zdalnej, która jest znana menedżerowi kolejek zdalnych. W tym przypadku *ObjectQMGrName* jest nazwą zdalnego menedżera kolejek.

Sprawdzenie poprawności zabezpieczeń wykonywane przez lokalny menedżer kolejek sprawdza, czy użytkownik jest uprawniony do wysyłania komunikatów do kolejki transmisji, wynikających z procesu rozstrzygania nazw.

W obu przypadkach:

- Lokalny menedżer kolejek nie wysyła komunikatów do zdalnego menedżera kolejek w celu sprawdzenia, czy użytkownik jest uprawniony do umieszczania komunikatów w kolejce.
- Gdy komunikat dociera do menedżera kolejek zdalnych, zdalny menedżer kolejek może go odrzucić, ponieważ użytkownik pochodzący z tego komunikatu nie jest autoryzowany.

Więcej informacji na ten temat zawierają pola *ObjectName* i *ObjectQMGrName* opisane w sekcji [“MQOD-deskryptor obiektu” na stronie 479](#) .

Obiekty

Zabezpieczenia

Poniższe uwagi odnoszą się do aspektów zabezpieczeń związanych z użyciem komendy MQOPEN.

Menedżer kolejek wykonuje sprawdzenia zabezpieczeń po wywołaniu wywołania MQOPEN w celu sprawdzenia, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma odpowiedni poziom uprawnień, zanim dostęp jest dozwolony. Sprawdzanie uprawnień jest wykonywane na podstawie nazwy otwieranego obiektu, a nie nazwy lub nazw, co spowodowało, że nazwa została rozwiązana.

Jeśli otwierany obiekt jest kolejką aliasową, która wskazuje obiekt tematu, menedżer kolejek wykonuje sprawdzenie zabezpieczeń w nazwie kolejki aliasowej, przed wykonaniem sprawdzenia zabezpieczeń tematu, tak jakby obiekt tematu był używany bezpośrednio.

Jeśli otwierany obiekt jest obiektem tematu, niezależnie od tego, czy jest on sam *ObjectName* , czy za pomocą *ObjectString* (z bazowaniem *ObjectName* lub bez), menedżer kolejek wykonuje sprawdzenie zabezpieczeń przy użyciu wynikowego łańcucha tematu, pobranego z obiektu tematu określonego w składce *ObjectName* i, jeśli jest to wymagane, konkatenując go z udostępnionym w programie *ObjectString* , a następnie znajdując najbliższy obiekt tematu w tym punkcie drzewa tematów lub znajdujący się powyżej tego punktu w celu wykonania sprawdzenia zabezpieczeń. Być może nie jest to ten sam obiekt tematu, który został określony w produkcie *ObjectName* .

Jeśli otwierany obiekt jest kolejką modelową, menedżer kolejek wykonuje pełne sprawdzenie zabezpieczeń zarówno dla nazwy kolejki modelowej, jak i nazwy kolejki dynamicznej, która jest tworzona. Jeśli wynikowa kolejka dynamiczna jest otwierana jawnie, dla nazwy kolejki dynamicznej wykonywane jest dalsze sprawdzanie zabezpieczeń zasobów.

W systemie z/OS menedżer kolejek wykonuje sprawdzenia zabezpieczeń tylko wtedy, gdy zabezpieczenia są włączone. Więcej informacji na temat sprawdzania zabezpieczeń znajduje się w sekcji [Konfigurowanie zabezpieczeń w systemie z/OS](#) .

Atrybuty

Poniższe uwagi odnoszą się do atrybutów.

Atrybuty obiektu mogą ulec zmianie, gdy aplikacja ma otwarty obiekt. W wielu przypadkach aplikacja tego nie zauważa, ale dla niektórych atrybutów menedżer kolejek oznacza uchwyt, który nie jest już poprawny. Są to następujące atrybuty:

- Dowolny atrybut, który ma wpływ na rozstrzyganie nazwy obiektu. Dotyczy to bez względu na używane opcje otwierania i obejmuje następujące elementy:

- Zmiana atrybutu *BaseQName* kolejki aliasowej, która jest otwarta.
- Zmiana atrybutu *TargetType* kolejki aliasowej, która jest otwarta.
- Zmiana atrybutów kolejki produktu *RemoteQName* lub *RemoteQMgrName* dla dowolnego uchwytu, który jest otwarty dla tej kolejki lub dla kolejki, która jest tłumaczona przez tę definicję jako alias menedżera kolejek.
- Każda zmiana, która powoduje, że aktualnie otwarty uchwyt kolejki zdalnej ma być rozstrzygany do innej kolejki transmisji lub w ogóle nie może być rozstrzygany. Na przykład może to być:
 - Zmiana atrybutu *XmitQName* w lokalnej definicji kolejki zdalnej, bez względu na to, czy definicja jest używana dla kolejki, czy dla aliasu menedżera kolejek.
 - (tylko w przypadku systemu z/OS) Zmiana wartości atrybutu menedżera kolejek produktu *IntraGroupqueuing* lub zmiana definicji współużytkowanej kolejki transmisji (SYSTEM.QSG.TRANSMIT.QUEUE) używany przez agenta IGQ.

Istnieje tylko jeden wyjątek: utworzenie nowej kolejki transmisji. Uchwyt, który mógł zostać rozwiązany do tej kolejki, był obecny podczas otwierania uchwytu, ale został rozstrzygnięty do domyślnej kolejki transmisji, nie jest on niepoprawny.

- Zmiana atrybutu menedżera kolejek produktu *DefXmitQName*. W tym przypadku wszystkie otwarte uchwyt, które zostały rozstrzygnięte do poprzednio nazwanej kolejki (która została rozstrzygnięta tylko dlatego, że była to domyślna kolejka transmisji) są oznaczone jako niepoprawne. Nie ma to wpływu na uchwyt, które zostały przetłumaczane na tę kolejkę z innych przyczyn.
- Atrybut kolejki *Shareability*, jeśli istnieją dwa lub więcej uchwytów, które obecnie zapewniają dostęp MQOO_INPUT_SHARED dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę. Jeśli tak, *wszystkie* uchwyt, które są otwarte dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, są oznaczone jako niepoprawne, niezależnie od otwartych opcji.

W systemie z/OS opisane wcześniej uchwyt są oznaczone jako niepoprawne, jeśli co najmniej jeden z uchwytów udostępnia do kolejki dostęp MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE.

- Atrybut kolejki *Usage* dla wszystkich uchwytów otwartych dla tej kolejki lub dla kolejki, która jest tłumaczona do tej kolejki, niezależnie od otwartych opcji.

Jeśli uchwyt jest oznaczony jako niepoprawny, wszystkie kolejne wywołania (inne niż MQCLOSE) korzystające z tego uchwytu nie powiodą się z kodem przyczyny MQRC_OBJECT_CHANGED. Aplikacja musi wywołać wywołanie MQCLOSE (przy użyciu oryginalnego uchwytu), a następnie ponownie otworzyć kolejkę. Wszystkie niezatwierdzone aktualizacje starego uchwytu z poprzednich pomyślnych wywołań nadal mogą być zatwierdzane lub wycofane, zgodnie z wymaganiami logiki aplikacji.

Jeśli zmiana atrybutu powoduje, że ma to nastąpić, należy użyć specjalnej wersji wymuszonej wywołania.

Wywołanie C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
        &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQOD      ObjDesc;    /* Object descriptor */
MQLONG    Options;    /* Options that control the action of MQOPEN */
MQHOBJ     Hobj;      /* Object handle */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN      PIC S9(9) BINARY.
** Object descriptor
01 OBJDESC.
   COPY CMQODV.
** Options that control the action of MQOPEN
01 OPTIONS    PIC S9(9) BINARY.
** Object handle
01 HOBJ       PIC S9(9) BINARY.
** Completion code
01 COMPCODE   PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON     PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl ObjDesc    like MQOD;    /* Object descriptor */
dcl Options    fixed bin(31); /* Options that control the action of
                               MQOPEN */
dcl Hobj       fixed bin(31); /* Object handle */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN      DS      F Connection handle
OBJDESC    CMQODA   , Object descriptor
OPTIONS    DS      F Options that control the action of MQOPEN
HOBJ       DS      F Object handle
COMPCODE   DS      F Completion code
REASON     DS      F Reason code qualifying COMPCODE
```

Wywołanie języka Visual Basic

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn      As Long 'Connection handle'
Dim ObjDesc    As Long '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-umieszczanie komunikatu

Wywołanie MQPUT umieszcza komunikat w kolejce lub na liście dystrybucyjnej albo w temacie. Kolejka, lista dystrybucyjna lub temat muszą być już otwarte.

Składnia

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Hobj

Typ: MQHOBJ-wejście

Ten uchwyt reprezentuje kolejkę, do której dodawany jest komunikat, lub temat, do którego komunikat jest publikowany. Wartość *Hobj* została zwrócona przez poprzednie wywołanie MQOPEN, które określiło opcję MQOO_OUTPUT.

MsgDesc

Typ: MQMD-input/output

Ta struktura opisuje atrybuty wysłanego komunikatu i otrzymuje informacje na temat komunikatu po zakończeniu żądania umieszczenia. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, dane komunikatu można poprzezzyć strukturą MQMDE, aby określić wartości dla pól istniejących w deskrypcyjnie MQMD w wersji version-2, ale nie w wersji version-1. Pole *Format* w strukturze MQMD musi być ustawione na wartość MQFMT_MD_EXTENSION, aby wskazać, że jest ona obecna w produkcie MQMDE. Więcej szczegółów na ten temat zawiera sekcja [“MQMDE-Rozszerzenie deskryptora komunikatu”](#) na stronie 471.

Aplikacja nie musi udostępniać struktury MQMD, jeśli poprawny uchwyt komunikatu jest dostępny w polach *OriginalMsg* lub *NewMsgUchwyt* struktury MQPMO. Jeśli w jednym z tych pól nie zostanie podana żadna wartość, deskryptor komunikatu jest przyjmowany z deskryptora powiązanego z uchwytami komunikatów.

Jeśli używany jest lub planowane jest użycie wyjść funkcji API, zaleca się, aby jawnie podać strukturę MQMD i nie używać deskryptorów komunikatów powiązanych z uchwytami komunikatów. Jest to spowodowane tym, że wyjście funkcji API powiązane z wywołaniem MQPUT lub MQPUT1 nie może sprawdzić, które wartości MQMD są używane przez menedżer kolejek w celu zakończenia żądania MQPUT lub MQPUT1.

OperacjePutMsg

Typ: MQPMO-input/output

Szczegółowe informacje na ten temat zawiera sekcja [“MQPMO-Put-message, opcje”](#) na stronie 500.

BufferLength

Typ: MQLONG-wejście

Długość komunikatu w produkcie *Buffer*. Wartość zero jest poprawna i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit dla *BufferLength* zależy od różnych czynników:

- Jeśli miejsce docelowe jest kolejką lokalną lub jest tłumaczone na kolejkę lokalną, górna granica zależy od tego, czy:

- Lokalny menedżer kolejek obsługuje segmentację.
- Aplikacja wysyłający określa flagę, która umożliwia menedżerowi kolejek segmentowanie komunikatu. Ta opcja ma wartość MQMF_SEGMENTATION_ALLOWED i może zostać określona w deskryptorach MQMD w wersji version-2 lub w produkcie MQMDE używanym z produktem MQMD w wersji version-1.

Jeśli oba te warunki zostaną spełnione, *BufferLength* nie może przekroczyć 999 999 999 minus wartość pola *Offset* w deskrypcyjnie MQMD. Najdłuższy komunikat logiczny, który może zostać umieszczony, wynosi 999 999 999 bajtów (gdy *Offset* jest zerem). Jednak ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w których aplikacja jest uruchomiona, może spowodować obniżenie limitu.

Jeśli jeden lub oba poprzednie warunki nie są spełnione, *BufferLength* nie może przekroczyć mniejszej wartości atrybutu *MaxMsgLength* kolejki i atrybutu *MaxMsgLength* menedżera kolejek.

- Jeśli miejsce docelowe jest kolejką zdalną lub jest tłumaczone na kolejkę zdalną, mają zastosowanie warunki dla kolejek lokalnych, *ale w każdym menedżerze kolejek, przez który musi przejść komunikat, aby dotrzeć do kolejki docelowej*; w szczególności:

1. Lokalna kolejka transmisji używana do tymczasowego przechowywania komunikatu w lokalnym menedżerze kolejek
2. Pośrednie kolejki transmisji (jeśli istnieją) używane do przechowywania komunikatu w menedżerach kolejek na trasie między lokalnymi i docelowymi menedżerami kolejek
3. Kolejka docelowa w docelowym menedżerze kolejek

Najdłuższy komunikat, który może zostać umieszczony, jest zarządzany przez najbardziej restrykcyjne dla tych kolejek i menedżerów kolejek.

Gdy komunikat znajduje się w kolejce transmisji, dodatkowe informacje znajdują się wraz z danymi komunikatu, co zmniejsza ilość danych aplikacji, które mogą być przenoszone. W tej sytuacji odejmij wartość MQ_MSG_HEADER_LENGTH w bajtach z wartości *MaxMsgLength* kolejek transmisji podczas określania limitu dla *BufferLength*.

Uwaga: Tylko niepowodzenie w zgodności z warunkiem 1 może być diagnozowane synchronicznie (z kodem przyczyny MQRC_MSG_TOO_BIG_FOR_Q lub MQRC_MSG_TOO_BIG_FOR_Q_MGR) po umieszczeniu komunikacie. Jeśli warunki 2 lub 3 nie są spełnione, komunikat zostanie przekierowany do kolejki niedostarczonych komunikatów (niedostarczonych komunikatów), albo w pośrednim menedżerze kolejek, albo w docelowym menedżerze kolejek. Jeśli tak się stanie, zostanie wygenerowany komunikat raportu, o ile został on zażądany przez nadawcę.

Buforuj

Typ: MQBYTEExBufferDługość-wejście

Jest to bufor zawierający dane aplikacji, które mają zostać wysłane. Bufor musi być wyrównany na granicy odpowiedniej do charakteru danych w komunikacie. 4-bajtowe wyrównanie jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajтового.

Jeśli *Buffer* zawiera dane znakowe lub numeryczne, ustaw wartości pól *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* na wartości odpowiednie dla danych. Umożliwia to odbiornikowi komunikatu przekształcenie danych (jeśli to konieczne) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek (nadawanego przez atrybut menedżera kolejek produktu *CodedCharSetId* i atrybut MQENC_NATIVE).

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void; adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *BufferLength* ma wartość zero, *Buffer* nie jest przywołany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') Grupa komunikatów nie została zakończona.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') Komunikat logiczny nie został zakończony.

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889 ') Niespójna specyfikacja trwałości.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

MQRC_PRIORITY_PRZEKRACZA_MAKSIMUM

(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838 ') W deskrytorze komunikatu nie rozpoznano opcji raportu.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ALIAS_TARGTYPE_CHANGED

(2480, X'09B0') Typ celu subskrypcji został zmieniony z kolejki na obiekt tematu.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BACKED_OUT

(2003, X'7D3') Wytworzona jednostka pracy.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'7D4') Parametr buforu nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

WYWOŁANIE mqrc_call_przerwane

(2549, X'9F5') Komenda MQPUT lub MQCMIT została przerwana i przetwarzanie ponownego połączenia nie może ponownie nawiązać określonego wyniku.

MQRC_CF_NOT_AVAILABLE

(2345, X' 929 ') Obiekt sprzęgający nie jest dostępny.

MQRC_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X' 945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CFGR_ERROR,

(2416, X' 970 ') Struktura parametru grupy PCF MQCFGR w danych komunikatu nie jest poprawna.

BŁĄD MQRC_CFH_ERROR

(2235, X'8BB') Struktura nagłówka PCF nie jest poprawna.

MQRC_CFIF_ERROR

(2414, X'96E') Struktura parametru filtru liczby całkowitej PCF w danych komunikatu nie jest poprawna.

MQRC_CFIL_ERROR,

(2236, X'8BC') Struktura parametru listy całkowitej PCF lub struktura parametru listy całkowitej PCIF*64 nie jest poprawna.

BŁĄD MQRC_CFIN_ERROR

(2237, X'8BD') Struktura parametru liczby całkowitej PCF lub struktura parametru liczby całkowitej PCIF*64 nie jest poprawna.

BŁĄD MQRC_CFSF_ERROR

(2415, X'96F') Struktura parametru filtru łańcucha PCF w danych komunikatu nie jest poprawna.

BŁĄD MQRC_CFSL_ERROR

(2238, X'8BE') Struktura parametru listy łańcuchów PCF nie jest poprawna.

MQRC_CFST_ERROR,

(2239, X'8BF') Struktura parametru łańcucha PCF nie jest poprawna.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') Błąd zasobu klastra.

MQRC_COD_NOT_VALID_FOR_XCF_Q

(2106, X'83A') Opcja raportu COD nie jest poprawna dla kolejki XCF.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CONTENT_ERROR

2554 (X'09FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

MQRC_CONTEXT_HANDLE_ERROR

(2097, X'831 ') Uchwyt kolejki, o którym mowa, nie zapisuje kontekstu.

MQRC_CONTEXT_NOT_AVAILABLE

(2098, X'832 ') Kontekst niedostępny dla uchwytu kolejki, o którym mowa.

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') Parametr długości danych nie jest poprawny.

BŁĄD MQRC_DH_ERROR

(2135, X'857 ') Struktura nagłówka dystrybucji nie jest poprawna.

BŁĄD MQRC_DLH_ERROR

(2141, X'85D') Struktura nagłówka niewysłanych wiadomości nie jest poprawna.

BŁĄD MQRC_EPH_ERROR

(2420, X' 974 ') Struktura osadzonego PCF jest niepoprawna.

BŁĄD MQRC_EXPIRY_ERROR

(2013, X'7DD') Czas utraty ważności nie jest poprawny.

Błąd MQRC_FEEDBACK_ERROR

(2014, X'7DE') Kod sprzężenia zwrotnego jest niepoprawny.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

MQRC_GROUP_ID_ERROR

(2258, X'8D2') Identyfikator grupy nie jest poprawny.

MQRC_HANDLE_IN_USE_DLA_UOW

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HEADER_ERROR

(2142, X'85E') Struktura nagłówka MQ nie jest poprawna.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

BŁĄD MQRC_IIH_ERROR

(2148, X'864 ') Struktura nagłówka informacji IMS nie jest poprawna.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') Grupa komunikatów nie została zakończona.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') Komunikat logiczny nie został zakończony.

MQRC_INCONSISTENT_PERSISTENCE

(2185, X'889 ') Niespójna specyfikacja trwałości.

MQRC_INCONSISTENT_UOW

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_LOCAL_UOW_CONFLICT

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

Błąd MQRC_MD_ERROR

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

MQRC_MDE_ERROR

(2248, X'8C8') Rozszerzenie deskryptora komunikatu nie jest poprawne.

MQRC_MISSING_REPLY_TO_Q,

(2027, X'7EB') Brak odpowiedzi na kolejkę odpowiedzi lub MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH

(2332, X'91C') Dane komunikatu nie rozpoczynają się od MQWIH.

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') Opcje komunikatu nie są poprawne.

MQRC_MSG_SEQ_NUMBER_ERROR,

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') Długość komunikatu jest większa niż maksymalna dla kolejki.

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

MQRC_MSG_TYPE_ERROR (BŁĄD)

(2029, X'7ED') Typ komunikatu w deskrypcji komunikatu nie jest poprawny.

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

MQRC_NO_DESTINATIONS_AVAILABLE

(2270, X'8DE') Nie są dostępne żadne kolejki docelowe.

MQRC_NOT_OPEN_FOR_OUTPUT

(2039, X'7F7') Kolejka nie jest otwarta dla danych wyjściowych.

MQRC_NOT_OPEN_FOR_PASS_ALL

(2093, X'82D') Kolejka nie jest otwarta dla przekazywania wszystkich kontekstów.

MQRC_NOT_OPEN_FOR_PASS_IDENT

(2094, X'82E') Kolejka nie jest otwarta dla przekazywania kontekstu tożsamości.

MQRC_NOT_OPEN_FOR_SET_ALL

(2095, X'82F') Kolejka nie jest otwarta dla ustawiania całego kontekstu.

MQRC_NOT_OPEN_FOR_SET_IDENT

(2096, X'830 ') Kolejka nie jest otwarta dla ustawiania kontekstu tożsamości.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

BŁĄD MQRC_OFFSET_ERROR

(2251, X'8CB') Przesunięcie segmentu komunikatu nie jest poprawne.

MQRC_OPEN_FAILED

(2137, X'859 ') Obiekt nie został otwarty pomyślnie.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

MQRC_ORIGINAL_LENGTH_ERROR

(2252, X'8CC') Oryginalna długość nie jest poprawna.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

BŁĄD MQRC_PCF_ERROR

(2149, X'865 ') Konstrukcje PCF nie są poprawne.

Błąd MQRC_PERSISTENCE_ERROR

(2047, X'7FF') Trwałość nie jest poprawna.

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

BŁĄD MQRC_PMO_ERROR

(2173, X'87D') Struktura opcji put-message nie jest poprawna.

MQRC_PMO_RECORD_FLAGS_ERROR

(2158, X'86E') flagi zapisu komunikatów nie są poprawne.

MQRC_PRIORITY_ERROR

(2050, X'802 ') Priorytet komunikatu nie jest poprawny.

MQRC_PUBLICATION_FAILURE

(2502, X'9C6') Publikacja nie została dostarczona do żadnego z subskrybentów.

MQRC_PUT_INHIBITED

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki, dla kolejki, do której ta kolejka jest tłumaczona, lub tematu.

MQRC_PUT_MSG_RECORDS_ERROR,

(2159, X'86F') rekordów umieszczania komunikatów nie jest poprawna.

MQRC_PUT_NOT_ZACHOWANE

(2479, X'09AF') Publikacja nie może być zachowana

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_FULL

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

MQRC_RECONNECT_NIE POWIODŁO SIĘ

(2548, X'9F4') Po ponownym połączeniu wystąpił błąd podczas ponownego podłączenia uchwytów dla połączenia z możliwością ponownego połączenia.

BŁĄD MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Liczba obecnie niepoprawnych rekordów.

MQRC_REPORT_OPTIONS_ERROR,

(2061, X'80D') Opcje raportu w deskryptorze komunikatu nie są poprawne.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_RESPONSE_RECORDS_ERROR,

(2156, X'86C') Rekordy odpowiedzi nie są poprawne.

BŁĄD MQRC_RFH_ERROR

(2334, X'91E') Struktura MQRFH lub struktura MQRFH2 nie jest poprawna.

MQRC_RMH_ERROR

(2220, X'8AC') Struktura nagłówka komunikatu odwołania nie jest poprawna.

MQRC_SEGMENT_LENGTH_ZERO

(2253, X'8CD') Długość danych w segmencie komunikatów wynosi zero.

MQRC_SEGMENTS_NOT_SUPPORTED

(2365, X'93D') Segmenty nie są obsługiwane.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') Istnieje potencjalny subskrybent publikacji, ale menedżer kolejek nie może sprawdzić, czy publikacja ma zostać wysłana do subskrybenta.

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

MQRC_STORAGE_CLASS_ERROR

(2105, X'839 ') Błąd klasy pamięci.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

BŁĄD MQRC_TM_ERROR

(2265, X'8D9') Struktura komunikatu wyzwalacza nie jest poprawna.

BŁĄD MQRC_TMC_ERROR

(2191, X'88F') Struktura komunikatu wyzwalacza znaku nie jest poprawna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

BŁĄD MQRC_WIH_ERROR

(2333, X'91D') Struktura MQWIH nie jest poprawna.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

BŁĄD MQRC_XQH_ERROR

(2260, X'8D4') Struktura nagłówka kolejki transmisji nie jest poprawna.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Uwagi dotyczące użycia tematu

1. Następujące uwagi mają zastosowanie w przypadku korzystania z tematów:

- a. W przypadku użycia komendy MQPUT do publikowania komunikatów w temacie, w którym co najmniej jeden subskrybent tego tematu nie może zostać nadany publikacji z powodu problemu z kolejką subskrybentów (na przykład jest pełna), kod przyczyny zwrócony do wywołania MQPUT i zachowanie dostarczania zależy od ustawienia atrybutów PMSGDLV lub NPMMSGDLV w temacie TOPIC. Należy zwrócić uwagę na dostarczenie publikacji do kolejki niedostarczonych komunikatów, jeśli określono wartość MQRO_DEAD_LETTER_Q lub odrzuć komunikat po określeniu parametru MQRO_DISCARD_MSG, który jest uznawany za pomyślne dostarczenie komunikatu. Jeśli żadna z publikacji nie zostanie dostarczona, komenda MQPUT zwróci wartość MQRC_PUBLICATION_FAILURE. Może się to zdarzyć w następujących przypadkach:

- Komunikat jest publikowany w TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALL, a każda subskrypcja (trwała lub nie) ma kolejkę, która nie może odbierać publikacji.
- Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLDUR, a subskrypcja trwała ma kolejkę, która nie może odbierać publikacji.

Operacja MQPUT może zwracać wartość MQRC_NONE, nawet jeśli publikacje nie mogą być dostarczane do niektórych subskrybentów w następujących przypadkach:

- Komunikat jest publikowany w TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLAVAIL, a każda subskrypcja, trwała lub nie, ma kolejkę, która nie może odbierać publikacji.
- Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLDUR, a subskrypcja nietrwała ma kolejkę, która nie może odbierać publikacji.

Za pomocą atrybutu tematu USEDLC można określić, czy kolejka niedostarczonych komunikatów jest używana, gdy komunikaty publikowania nie mogą być dostarczane do odpowiedniej kolejki subskrybenta. Więcej informacji na temat użycia parametru USEDLC znajduje się w sekcji [DEFINE TOPIC\(DEFINIOWANIE TEMATU\)](#).

- Jeśli nie ma subskrybentów w używanym temacie, opublikowany komunikat nie jest wysyłany do żadnej kolejki i jest usuwany. Nie ma znaczenia, czy komunikat jest trwały, czy nietrwały, czy ma nieograniczony limit czasu utraty ważności, czy ma czas utraty ważności, ale nadal jest odrzucany, jeśli nie ma subskrybentów. Wyjątkiem jest sytuacja, w której komunikat ma zostać zachowany. W takim przypadku, mimo że nie jest on wysyłany do kolejek subskrybentów, zostanie on zapisany w temacie, który ma zostać dostarczony do nowych subskrypcji lub do wszystkich subskrybentów, którzy poproszli o zachowane publikacje za pomocą komendy MQSUBRQ.

MQPUT i MQPUT1

Można użyć zarówno wywołań MQPUT, jak i MQPUT1 w celu umieszczenia komunikatów w kolejce. W zależności od okoliczności używane jest wywołanie do użycia.

- Użyj wywołania MQPUT, aby umieścić wiele komunikatów w kolejce *ta sama*.

Wywołanie MQOPEN z opcją MQOO_OUTPUT jest wysyłane jako pierwsze, po którym następuje jedna lub większa liczba żądań MQPUT w celu dodania komunikatów do kolejki. W końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1.

- Użyj wywołania MQPUT1, aby umieścić tylko *jeden* komunikat w kolejce.

Wywołanie to enkapsuluje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wysłane.

Kolejki docelowe

Do korzystania z kolejek docelowych mają zastosowanie następujące uwagi:

- Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, kolejność tych komunikatów jest zachowywana, jeśli spełnione są szczegółowe warunki. Niektóre warunki mają zastosowanie zarówno do lokalnych, jak i zdalnych kolejek docelowych; inne warunki mają zastosowanie tylko do kolejek zdalnych miejsc docelowych.

Warunki, które mają zastosowanie do lokalnych i zdalnych kolejek docelowych

- Wszystkie wywołania MQPUT znajdują się w tej samej jednostce pracy lub żaden z nich nie znajduje się w obrębie jednostki pracy.

Należy pamiętać, że gdy komunikaty są umieszczane w określonej kolejce w ramach pojedynczej jednostki pracy, komunikaty z innych aplikacji mogą być przeplatane z kolejnością komunikatów w kolejce.

- Wszystkie wywołania MQPUT są wykonywane przy użyciu tego samego uchwytu obiektu *Hobj*.

W niektórych środowiskach kolejność komunikatów jest również zachowywana, gdy używane są różne uchwyty obiektów, jeśli wywołania są wykonywane z tej samej aplikacji. Znaczenie *tej samej aplikacji* jest określone przez środowisko:

- W systemie z/OS aplikacja jest następująca:
 - W przypadku produktu CICS zadanie CICS
 - W przypadku produktu IMS: zadanie
 - W przypadku zadania wsadowego z/OS zadanie
- W systemie IBM i aplikacja jest zadaniem.
- W systemach Windows i Systemy UNIX aplikacja jest wątkiem.
- Wszystkie komunikaty mają ten sam priorytet.
- Komunikaty nie są umieszczane w kolejce klastra z określoną wartością MQOO_BIND_NOT_FIXED (lub z wartością MQOO_BIND_AS_Q_DEF w momencie, gdy atrybut kolejki DefBind ma wartość MQBND_BIND_NOT_FIXED).

Dodatkowe warunki mające zastosowanie do kolejek zdalnych miejsc docelowych

- Istnieje tylko jedna ścieżka od wysyłającego menedżera kolejek do docelowego menedżera kolejek.

Jeśli niektóre komunikaty w sekwencji mogą znajdować się w innej ścieżce (na przykład z powodu rekonfiguracji, równoważenia ruchu lub wyboru ścieżki w zależności od wielkości komunikatu), nie można zagwarantować kolejności komunikatów w docelowym menedżerze kolejek.

- Komunikaty nie są tymczasowo umieszczane w kolejkach niedostarczonych komunikatów w menedżerach kolejek nadawczych, pośrednich i docelowych.

Jeśli co najmniej jeden komunikat jest tymczasowo umieszczany w kolejce niedostarczonych komunikatów (na przykład, ponieważ kolejka transmisji lub kolejka docelowa jest tymczasowo pełna), komunikaty mogą być odbierane w kolejce docelowej poza kolejnością.

- Komunikaty są albo wszystkie trwałe, albo wszystkie nietrwałe.

Jeśli kanał na trasie między menedżerami kolejek wysyłających i docelowych ma atrybut *NonPersistentMsgSpeed* ustawiony na wartość MQNPMS_FAST, komunikaty nietrwałe mogą przechodzić do przodu w stosunku do trwałych komunikatów, co powoduje, że porządek komunikatów trwałych względem nietrwałych komunikatów nie jest zachowany. Jednak kolejność komunikatów trwałych względem siebie oraz komunikatów nietrwałych, względem siebie wzajemnie, jest zachowywana.

Jeśli warunki te nie są spełnione, można użyć grup komunikatów w celu zachowania kolejności komunikatów, ale wymaga to zarówno aplikacji wysyłającej, jak i odbierającej w celu użycia obsługi grupowania komunikatów. Więcej informacji na temat grup komunikatów zawiera sekcja:

- [MQMD-pole MsgFlags](#)
- [MQPMO_LOGICAL_ORDER](#)
- [MQGMO_LOGICAL_ORDER](#)

Listy dystrybucyjne

Do korzystania z list dystrybucyjnych stosuje się następujące uwagi.

Listy dystrybucyjne są obsługiwane w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączone z tymi systemami.

1. Komunikaty można umieszczać na liście dystrybucyjnej przy użyciu produktu version-1 lub version-2 MQPMO. Jeśli używana jest wartość version-1 MQPMO (lub version-2 MQPMO z wartością

RecsPresent równą zero), wówczas aplikacja nie może udostępnić rekordów komunikatów lub rekordów odpowiedzi. Nie można zidentyfikować kolejek, w których występują błędy, jeśli komunikat został pomyślnie wysłany do niektórych kolejek na liście dystrybucyjnej, a nie do innych.

Jeśli aplikacja udostępnia rekordy komunikatów lub rekordy odpowiedzi, należy ustawić pole *Version* na wartość MQPMO_VERSION_2.

Można również użyć programu MQPMO version-2 do wysyłania komunikatów do jednej kolejki, która nie znajduje się na liście dystrybucyjnej, upewniając się, że parametr *RecsPresent* ma wartość zero.

2. Kod zakończenia i parametry kodu przyczyny są ustawione w następujący sposób:

- Jeśli wszystkie operacje umieszczania w kolejkach na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, to kod zakończenia i parametry kodu przyczyny zostaną ustawione w taki sposób, aby opisywać wspólny wynik. W tym przypadku nie są ustawione rekordy odpowiedzi MQRR (jeśli aplikacja jest udostępniana przez aplikację).

Na przykład, jeśli każde wykonanie powiedzie się, kod zakończenia i kod przyczyny są ustawiane na wartość MQCC_OK i MQRC_NONE; jeśli wszystkie operacje umieszczania nie powiodą się, ponieważ wszystkie kolejki są zablokowane dla operacji put, to parametry są ustawiane na wartość MQCC_FAILED i MQRC_PUT_INHIBITED.

- Jeśli operacje umieszczania w kolejkach na liście dystrybucyjnej nie wszystkie powiodą się lub nie powiodą się w ten sam sposób:
 - Parametr kodu zakończenia jest ustawiony na wartość MQCC_WARNING, jeśli co najmniej jedno zostało pomyślnie wykonane, a dla parametru MQCC_FAILED, jeśli wszystkie nie powiodły się.
 - Parametr kodu przyczyny jest ustawiony na wartość MQRC_MULTIPLE_UZASADNIENIE.
 - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.

Jeśli operacja put dla miejsca docelowego nie powiedzie się, ponieważ otwarcie dla tego miejsca docelowego nie powiodło się, pola w rekordzie odpowiedzi są ustawione na wartość MQCC_FAILED i MQRC_OPEN_FAILED; miejsce docelowe jest dołączone do produktu *InvalidDestCount*.

3. Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę lokalną, komunikat jest umieszczany w tej kolejce w normalnej formie (czyli nie jako komunikat z listą dystrybucyjną). Jeśli więcej niż jedno miejsce docelowe jest tłumaczone na tę samą kolejkę lokalną, w kolejce dla każdego z tych miejsc docelowych umieszczany jest jeden komunikat.

Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę zdalną, komunikat jest umieszczany w odpowiedniej kolejce transmisji. W przypadku, gdy kilka miejsc docelowych jest rozstrzyganych w tej samej kolejce transmisji, w kolejce transmisji może zostać umieszczony pojedynczy komunikat z listą dystrybucyjną zawierający te miejsca docelowe, nawet jeśli te miejsca docelowe nie były umieszczone obok listy miejsc docelowych udostępnionych przez aplikację. Można to jednak zrobić tylko wtedy, gdy kolejka transmisji obsługuje komunikaty listy dystrybucyjnej (patrz sekcja [DistLists](#)).

Jeśli kolejka transmisji nie obsługuje list dystrybucyjnych, jedna kopia komunikatu w zwykłej formie jest umieszczana w kolejce transmisji dla każdego miejsca docelowego, które korzysta z tej kolejki transmisji.

Jeśli lista dystrybucyjna z danymi komunikatu aplikacji jest zbyt duża dla kolejki transmisji, komunikat listy dystrybucyjnej jest dzielony na mniejsze komunikaty listy dystrybucyjnej, z których każda zawiera mniej miejsc docelowych. Jeśli dane komunikatu aplikacji tylko wpisują się do kolejki, komunikaty listy dystrybucyjnej nie mogą być używane w ogóle, a menedżer kolejek generuje jedną kopię komunikatu w postaci normalnej dla każdego miejsca docelowego, które korzysta z tej kolejki transmisji.

Jeśli różne miejsca docelowe mają inny priorytet komunikatu lub trwałość komunikatu (może to wystąpić, gdy aplikacja określa wartość MQPRI_PRIORITY_AS_Q_DEF lub MQPER_PERSISTENCE_AS_Q_DEF), komunikaty nie są przechowywane w tej samej komunikacie listy dystrybucyjnej. Zamiast tego menedżer kolejek generuje tyle komunikatów listy dystrybucyjnej, które są niezbędne do uwzględnienia różnych wartości priorytetu i trwałości.

4. Umieszczenie na liście dystrybucyjnej może spowodować:

- Pojedynczy komunikat z listą dystrybucyjną, lub
- Liczba mniejszych komunikatów listy dystrybucyjnej, lub
- Mieszanina komunikatów listy dystrybucyjnej i zwykłych komunikatów, lub
- Tylko komunikaty normalne.

To, które z powyższych występuje, zależy od tego, czy:

- Miejsca docelowe na liście są lokalne, zdalne lub mieszane.
- Miejsca docelowe mają ten sam priorytet komunikatu i trwałość komunikatu.
- Kolejki transmisji mogą zawierać komunikaty listy dystrybucyjnej.
- Maksymalna długość kolejek transmisji jest wystarczająco duża, aby pomieścić komunikat w postaci listy dystrybucyjnej.

Jednak niezależnie od tego, który z powyższych zdarzeń występuje, każdy komunikat *fizyczny* (czyli każdy komunikat normalny lub komunikat listy dystrybucyjnej będący wynikiem umieszczenia) jest wyświetlany jako tylko *jeden* komunikat, gdy:

- Sprawdzanie, czy aplikacja przekroczyła dozwoloną maksymalną liczbę komunikatów w jednostce pracy (patrz atrybut menedżera kolejek *MaxUncommittedMsgs*).
 - Sprawdzanie, czy warunki wyzwiania są spełnione.
 - Zwiększ głębokość kolejki i sprawdź, czy maksymalna głębokość kolejki została przekroczona.
5. Każda zmiana w definicjach kolejek, które spowodowałyby, że uchwyt stał się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana ścieżki rozdzielczej), nie powoduje, że uchwyt listy dystrybucyjnej staje się niepoprawny. Jednak powoduje to niepowodzenie tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnych wywoławczych wywołania MQPUT.

Nagłówki

Jeśli na początku danych komunikatu aplikacji zostanie umieszczony komunikat zawierający jedną lub większą liczbę struktur nagłówka produktu IBM MQ, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka w celu sprawdzenia, czy są one poprawne. Jeśli menedżer kolejek wykryje błąd, wywołanie nie powiedzie się i zostanie zwrócony odpowiedni kod przyczyny. Przeprowadzone kontrole różnią się w zależności od obecnych struktur, które są obecne:

- Sprawdzenia są wykonywane tylko wtedy, gdy w wywołaniu MQPUT lub MQPUT1 jest używany deskryptor MQMD w wersji version-2 lub późniejszej. Sprawdzanie nie jest wykonywane, jeśli używany jest deskryptor MQMD w wersji version-1, nawet jeśli na początku danych komunikatu jest obecny deskryptor MQMDE.
- Struktury, które nie są obsługiwane przez lokalny menedżer kolejek i struktury po pierwszym komunikacie MQDLH w komunikacie, nie są sprawdzane.
- Poprawność struktur MQDH i MQMDE jest sprawdzana w całości przez menedżer kolejek.
- Poprawność innych struktur jest sprawdzana częściowo przez menedżer kolejek (nie wszystkie pola są sprawdzane).

Ogólne sprawdzenia wykonywane przez menedżera kolejek obejmują następujące elementy:

- Pole *StrucId* musi być poprawne.
- Pole *Version* musi być poprawne.
- W polu *StrucLength* należy podać wartość, która jest wystarczająco duża, aby uwzględnić strukturę powiększoną o dowolne dane o zmiennej długości, które są częścią struktury.
- Pole *CodedCharSetId* nie może być zerowe lub wartość ujemna, która nie jest poprawna (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR i MQCCSI_UNDEFINED *nie* jest poprawna w większości struktur nagłówka produktu IBM MQ).
- Parametr *BufferLength* w wywołaniu musi określać wartość, która jest wystarczająco duża, aby uwzględnić strukturę (struktura nie może wykraczać poza koniec komunikatu).

Oprócz ogólnych kontroli struktur, muszą być spełnione następujące warunki:

- Suma długości struktur w komunikacie PCF musi być równa długości określonej za pomocą parametru *BufferLength* w wywołaniu MQPUT lub MQPUT1. Komunikat PCF to komunikat o formacie nazwy MQFMT_ADMIN, MQFMT_EVENT lub MQFMT_PCF.
- Struktura IBM MQ nie może zostać obciążona, z wyjątkiem sytuacji, w których dozwolone są obciążone struktury:
 - Komunikaty, które są komunikatami raportu.
 - Komunikaty PCF.
 - Komunikaty zawierające strukturę MQDLH. (Struktury *następujące* pierwsze wywołanie MQDLH może zostać obciążone; struktury poprzedzające obiekt MQDLH nie mogą być obciążane).
- Struktura IBM MQ nie może być podzielona na dwa lub więcej segmentów; struktura musi być zawarta w całości w jednym segmencie.

Buforuj

W przypadku języka programowania Visual Basic, zastosowanie mają następujące punkty:

- Jeśli wielkość parametru *Buffer* jest mniejsza niż długość określona przez parametr *BufferLength*, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_BUFFER_LENGTH_ERROR.
- Parametr *Buffer* jest zadeklarowany jako typ String. Jeśli dane, które mają być umieszczone w kolejce, nie są typu String, należy użyć wywołania MQPUTAny w miejsce MQPUT.

Wywołanie MQPUTAny ma takie same parametry, jak wywołanie MQPUT, z wyjątkiem tego, że parametr *Buffer* jest zadeklarowany jako typ Any, co pozwala na umieszczanie w kolejce dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić *Buffer*, aby upewnić się, że wielkość ta wynosi co najmniej *BufferLength* bajtów.

Wywołanie C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,  
       &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;          /* Connection handle */  
MQHOBJ   Hobj;           /* Object handle */  
MQMD     MsgDesc;        /* Message descriptor */  
MQPMO    PutMsgOpts;     /* Options that control the action of MQPUT */  
MQLONG   BufferLength;    /* Length of the message in Buffer */  
MQBYTE   Buffer[n];       /* Message data */  
MQLONG   CompCode;       /* Completion code */  
MQLONG   Reason;         /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,  
                  BUFFER, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN      PIC S9(9) BINARY.  
** Object handle  
01 HOBJ       PIC S9(9) BINARY.  
** Message descriptor  
01 MSGDESC.  
   COPY CMQMDV.  
** Options that control the action of MQPUT
```

```

01 PUTMSGOPTS.
   COPY CMQPMOV.
** Length of the message in BUFFER
01 BUFFERLENGTH PIC S9(9) BINARY.
** Message data
01 BUFFER       PIC X(n).
** Completion code
01 COMPCODE     PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON       PIC S9(9) BINARY.

```

Wywołanie PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```

dcl Hconn      fixed bin(31); /* Connection handle */
dcl Hobj       fixed bin(31); /* Object handle */
dcl MsgDesc    like MQMD;    /* Message descriptor */
dcl PutMsgOpts like MQPMO;    /* Options that control the action of
                               MQPUT */
dcl BufferLength fixed bin(31); /* Length of the message in Buffer */
dcl Buffer      char(n);       /* Message data */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */

```

Wywołanie High Level Assembler

```
CALL MQPUT, (HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
            BUFFER,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
MSGDESC	CMQMDA	,	Message descriptor
PUTMSGOPTS	CMQPMOA	,	Options that control the action of MQPUT
BUFFERLENGTH	DS	F	Length of the message in BUFFER
BUFFER	DS	CL(n)	Message data
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
Reason
```

Zadeklaruj parametry w następujący sposób:

```

Dim Hconn      As Long 'Connection handle'
Dim Hobj       As Long 'Object handle'
Dim MsgDesc    As MQMD 'Message descriptor'
Dim PutMsgOpts As MQPMO 'Options that control the action of MQPUT'
Dim BufferLength As Long 'Length of the message in Buffer'
Dim Buffer      As String 'Message data'
Dim CompCode   As Long 'Completion code'
Dim Reason     As Long 'Reason code qualifying CompCode'

```

MQPUT1 -Umieść jeden komunikat

Wywołanie MQPUT1 umieszcza jeden komunikat w kolejce lub na liście dystrybucyjnej albo w temacie.

Kolejka, lista dystrybucyjna lub temat nie muszą być otwarte.

Składnia

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

ObjDesc

Typ: MQOD-input/output

Jest to struktura identyfikująca kolejkę, do której komunikat jest dodawany, lub temat, do którego komunikat jest publikowany. Szczegółowe informacje na ten temat zawiera sekcja [“MQOD-deskryptor obiektu”](#) na stronie 479.

Jeśli struktura jest kolejką, użytkownik musi mieć uprawnienia do otwarcia kolejki dla danych wyjściowych. Kolejka modelowa **nie** musi być kolejką modelową.

MsgDesc

Typ: MQMD-input/output

Ta struktura opisuje atrybuty wysyłanego komunikatu i otrzymuje informację zwrotną po zakończeniu żądania umieszczenia. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1 , dane komunikatu można poprzeżyć strukturą MQMDE, aby określić wartości dla pól istniejących w deskrypcyjnie MQMD w wersji version-2 , ale nie w wersji version-1. Ustaw pole *Format* w strukturze MQMD na wartość MQFMT_MD_EXTENSION, aby wskazać, że jest obecna tabela MQMDE. Więcej szczegółów na ten temat zawiera sekcja [“MQMDE-Rozszerzenie deskryptora komunikatu”](#) na stronie 471.

Aplikacja nie musi udostępniać struktury MQMD, jeśli poprawny uchwyt komunikatu jest dostępny w polu *MsgHandle* struktury MQGMO lub w polach *OriginalMsgHandle* lub *NewMsgHandle* w strukturze MQPMO. Jeśli w jednym z tych pól nie zostanie podana żadna wartość, deskryptor komunikatu jest przyjmowany z deskryptora powiązanego z uchwytami komunikatów.

OperacjePutMsg

Typ: MQPMO-input/output

Szczegółowe informacje na ten temat zawiera sekcja [“MQPMO-Put-message, opcje”](#) na stronie 500.

BufferLength

Typ: MQLONG-wejście

Długość komunikatu w produkcie *Buffer*. Wartość zero jest poprawna i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit zależy od różnych czynników. Opis parametru *BufferLength* można znaleźć w sekcji [“MQPUT-umieszczanie komunikatu”](#) na stronie 760 .

Buforuj

Typ: MQBYTEExBufferDługość-wejście

Jest to bufor zawierający dane komunikatu aplikacji, które mają zostać wysłane. Wyrównaj bufor na granicy, odpowiedni do charakteru danych w komunikacie. 4-bajtowe wyrównanie jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajтового.

Jeśli *Buffer* zawiera dane znakowe lub numeryczne, ustaw wartości pól *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* na wartości odpowiednie dla danych. Umożliwia to odbiornikowi komunikatu przekształcenie danych (jeśli to konieczne) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT1 muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek (nadawanego przez atrybut menedżera kolejek produktu *CodedCharSetId* i atrybut MQENC_NATIVE).

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void; adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *BufferLength* ma wartość zero, *Buffer* nie jest przywołany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

MQRC_INCOMPLETE_GROUP

(2241, X'8C1') Grupa komunikatów nie została zakończona.

MQRC_INCOMPLETE_MSG

(2242, X'8C2') Komunikat logiczny nie został zakończony.

MQRC_PRIORITY_PRZEKRACZA_MAKSIMUM

(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

MQRC_UNKNOWN_REPORT_OPTION

(2104, X'838 ') Opcje raportu w deskrypcji komunikatu nie zostały rozpoznane.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ALIAS_BASE_Q_TYPE_ERROR

(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BACKED_OUT

(2003, X'7D3') Wytworzona jednostka pracy.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'7D4') Parametr buforu nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_NOT_AVAILABLE

(2345, X' 929 ') narzędzie sprzęgające nie jest dostępne.

MQRC_CF_STRUC_AUTH_FAILED

(2348, X'92C') Sprawdzanie autoryzacji struktury CF (Coupling-Facility) nie powiodło się.

MQRC_CF_STRUC_ERROR

(2349, X'92D') Struktura CF (Coupling-Facility) jest niepoprawna.

MQRC_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X' 945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') Lista struktury narzędzia CF-nagłówków w użyciu.

MQRC_CFGR_ERROR,

(2416, X' 970 ') Struktura parametru grupy PCF MQCFGR w danych komunikatu nie jest poprawna.

BŁĄD MQRC_CFH_ERROR

(2235, X'8BB') Struktura nagłówka PCF nie jest poprawna.

MQRC_CFIF_ERROR

(2414, X'96E') Struktura parametru filtru liczby całkowitej PCF w danych komunikatu nie jest poprawna.

MQRC_CFIL_ERROR,

(2236, X'8BC') Struktura parametru listy całkowitej PCF lub struktura parametru listy całkowitej PCIF*64 nie jest poprawna.

BŁĄD MQRC_CFIN_ERROR

(2237, X'8BD') Struktura parametru liczby całkowitej PCF lub struktura parametru liczby całkowitej PCIF*64 nie jest poprawna.

BŁĄD MQRC_CFSF_ERROR

(2415, X'96F') Struktura parametru filtru łańcucha PCF w danych komunikatu nie jest poprawna.

BŁĄD MQRC_CFSL_ERROR

(2238, X'8BE') Struktura parametru listy łańcuchów PCF nie jest poprawna.

MQRC_CFST_ERROR,

(2239, X'8BF') Struktura parametru łańcucha PCF nie jest poprawna.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CLUSTER_EXIT_ERROR

(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

MQRC_CLUSTER_RESOURCE_ERROR

(2269, X'8DD') Błąd zasobu klastra.

MQRC_COD_NOT_VALID_FOR_XCF_Q

(2106, X'83A') Opcja raportu COD nie jest poprawna dla kolejki XCF.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING

(2202, X'89A') Połączenie wygaszające.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CONTENT_ERROR

2554 (X'09FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat może zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

MQRC_CONTEXT_HANDLE_ERROR

(2097, X'831 ') Uchwyt kolejki, o którym mowa, nie zapisuje kontekstu.

MQRC_CONTEXT_NOT_AVAILABLE

(2098, X'832 ') Kontekst niedostępny dla uchwytu kolejki, o którym mowa.

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') Parametr długości danych nie jest poprawny.

MQRC_DB2_NOT_AVAILABLE

(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

MQRC_DEF_XMIT_Q_TYPE_ERROR

(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

MQRC_DEF_XMIT_Q_USAGE_ERROR

(2199, X'897 ') Domyślny błąd wykorzystania kolejki transmisji.

BŁĄD MQRC_DH_ERROR

(2135, X'857 ') Struktura nagłówka dystrybucji nie jest poprawna.

BŁĄD MQRC_DLH_ERROR

(2141, X'85D') Struktura nagłówka niewysłanych wiadomości nie jest poprawna.

BŁĄD MQRC_EPH_ERROR

(2420, X' 974 ') Struktura osadzonego PCF jest niepoprawna.

BŁĄD MQRC_EXPIRY_ERROR

(2013, X'7DD') Czas utraty ważności nie jest poprawny.

Błąd MQRC_FEEDBACK_ERROR

(2014, X'7DE') Kod sprzężenia zwrotnego jest niepoprawny.

MQRC_GLOBAL_UOW_CONFLICT

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

MQRC_GROUP_ID_ERROR

(2258, X'8D2') Identyfikator grupy nie jest poprawny.

MQRC_HANDLE_IN_USE_DLA_UOW

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

MQRC_HANDLE_NOT_AVAILABLE

(2017, X'7E1') Nie ma więcej dostępnych uchwytów.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HEADER_ERROR

(2142, X'85E') Struktura nagłówka IBM MQ nie jest poprawna.

BŁĄD MQRC_IIH_ERROR

(2148, X'864 ') Struktura nagłówka informacji IMS nie jest poprawna.

MQRC_LOCAL_UOW_CONFLICT

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

Błąd MQRC_MD_ERROR

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

MQRC_MDE_ERROR

(2248, X'8C8') Rozszerzenie deskryptora komunikatu nie jest poprawne.

MQRC_MISSING_REPLY_TO_Q,

(2027, X'7EB') Brak odpowiedzi na kolejkę odpowiedzi.

MQRC_MISSING_WIH

(2332, X'91C') Dane komunikatu nie rozpoczynają się od MQWIH.

MQRC_MSG_FLAGS_ERROR

(2249, X'8C9') Opcje komunikatu nie są poprawne.

MQRC_MSG_SEQ_NUMBER_ERROR,

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

MQRC_MSG_TOO_BIG_FOR_Q

(2030, X'7EE') Długość komunikatu jest większa niż maksymalna dla kolejki.

MQRC_MSG_TOO_BIG_FOR_Q_MGR

(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

MQRC_MSG_TYPE_ERROR (BŁĄD)

(2029, X'7ED') Typ komunikatu w deskrypcie komunikatu nie jest poprawny.

MQRC_MULTIPLE_POWODY

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

MQRC_NO_DESTINATIONS_AVAILABLE

(2270, X'8DE') Nie są dostępne żadne kolejki docelowe.

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

MQRC_OBJECT_IN_USE

(2042, X'7FA') Obiekt jest już otwarty z opcjami powodującymi konflikt.

MQRC_OBJECT_LEVEL_NIEZGODNY

(2360, X' 938 ') Poziom obiektu nie jest kompatybilny.

MQRC_OBJECT_NAME_ERROR

(2152, X'868 ') Nazwa obiektu nie jest poprawna.

MQRC_OBJECT_NOT_UNIQUE

(2343, X' 927 ') Obiekt nie jest unikalny.

Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR

(2153, X'869 ') Nazwa menedżera kolejek obiektu nie jest poprawna.

MQRC_OBJECT_RECORDS_ERROR

(2155, X'86B') Rekordy obiektów nie są poprawne.

MQRC_OBJECT_TYPE_ERROR

(2043, X'7FB') Typ obiektu nie jest poprawny.

BŁĄD MQRC_OD_ERROR

(2044, X'7FC') Struktura deskryptora obiektu nie jest poprawna.

BŁĄD MQRC_OFFSET_ERROR

(2251, X'8CB') Przesunięcie segmentu komunikatu nie jest poprawne.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

MQRC_ORIGINAL_LENGTH_ERROR

(2252, X'8CC') Oryginalna długość nie jest poprawna.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

BŁĄD MQRC_PCF_ERROR

(2149, X'865 ') Konstrukcje PCF nie są poprawne.

Błąd MQRC_PERSISTENCE_ERROR

(2047, X'7FF') Trwałość nie jest poprawna.

MQRC_PERSISTENT_NOT_ALLOWED

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

BŁĄD MQRC_PMO_ERROR

(2173, X'87D') Struktura opcji put-message nie jest poprawna.

MQRC_PMO_RECORD_FLAGS_ERROR

(2158, X'86E') flagi zapisu komunikatów nie są poprawne.

MQRC_PRIORITY_ERROR

(2050, X'802 ') Priorytet komunikatu nie jest poprawny.

MQRC_PUBLICATION_FAILURE

(2502, X'9C6') Publikacja nie została dostarczona do żadnego z subskrybentów.

MQRC_PUT_INHIBITED

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki.

MQRC_PUT_MSG_RECORDS_ERROR,

(2159, X'86F') rekordów umieszczania komunikatów nie jest poprawna.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_FULL

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR QUIESCING,

(2161, X'871 ') Menedżer kolejek jest wygaszany.

MQRC_Q_MGR_ZATRZYMOWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_SPACE_NOT_AVAILABLE

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

MQRC_Q_TYPE_ERROR

(2057, X'809 ') Typ kolejki nie jest poprawny.

BŁĄD MQRC_RECS_PRESENT_ERROR

(2154, X'86A') Liczba obecnie niepoprawnych rekordów.

MQRC_REMOTE_Q_NAME_ERROR

(2184, X'888 ') Nazwa zdalnej kolejki nie jest poprawna.

MQRC_REPORT_OPTIONS_ERROR,

(2061, X'80D') Opcje raportu w deskrytorze komunikatu nie są poprawne.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_RESPONSE_RECORDS_ERROR,

(2156, X'86C') Rekordy odpowiedzi nie są poprawne.

BŁĄD MQRC_RFH_ERROR

(2334, X'91E') Struktura MQRFH lub struktura MQRFH2 nie jest poprawna.

MQRC_RMH_ERROR

(2220, X'8AC') Struktura nagłówka komunikatu odwołania nie jest poprawna.

MQRC_SECURITY_ERROR,

(2063, X'80F') Wystąpił błąd zabezpieczeń.

MQRC_SEGMENT_LENGTH_ZERO

(2253, X'8CD') Długość danych w segmencie komunikatów wynosi zero.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') Istnieje potencjalny subskrybent publikacji, ale menedżer kolejek nie może sprawdzić, czy publikacja ma zostać wysłana do subskrybenta.

MQRC_STOPPED_BY_CLUSTER_EXIT

(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

MQRC_STORAGE_CLASS_ERROR

(2105, X'839 ') Błąd klasy pamięci.

MQRC_STORAGE_MEDIUM_FULL

(2192, X'890 ') Zewnętrzny nośnik pamięci jest pełny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

MQRC_SYNCPOINT_NOT_AVAILABLE

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

BŁĄD MQRC_TM_ERROR

(2265, X'8D9') Struktura komunikatu wyzwalacza nie jest poprawna.

BŁĄD MQRC_TMC_ERROR

(2191, X'88F') Struktura komunikatu wyzwalacza znaku nie jest poprawna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UNKNOWN_ALIAS_BASE_Q

(2082, X'822 ') Nieznana kolejka podstawowa aliasu.

MQRC_UNKNOWN_DEF_XMIT_Q

(2197, X'895 ') Nieznana domyślna kolejka transmisji.

MQRC_UNKNOWN_OBJECT_NAME

(2085, X'825 ') Nieznana nazwa obiektu.

MQRC_UNKNOWN_OBJECT_Q_MGR

(2086, X'826 ') Nieznany menedżer kolejek obiektów.

MQRC_UNKNOWN_REMOTE_Q_MGR

(2087, X'827 ') Nieznany zdalny menedżer kolejek.

MQRC_UNKNOWN_XMIT_Q

(2196, X'894 ') Nieznana kolejka transmisji.

MQRC_UOW_ENLISTMENT_ERROR

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

MQRC_UOW_MIX_NOT_SUPPORTED

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

MQRC_UOW_NOT_AVAILABLE

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

BŁĄD MQRC_WIH_ERROR

(2333, X'91D') Struktura MQWIH nie jest poprawna.

MQRC_WRONG_CF_LEVEL

(2366, X'93E') Struktura CF (Coupling-Facility) jest poziomem błędnym.

MQRC_WRONG_MD_VERSION

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

MQRC_XMIT_Q_TYPE_ERROR

(2091, X'82B') Kolejka transmisji nie jest lokalna.

MQRC_XMIT_Q_USAGE_ERROR,

(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

BŁĄD MQRC_XQH_ERROR

(2260, X'8D4') Struktura nagłówka kolejki transmisji nie jest poprawna.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. Wywołanie w celu użycia zależy od okoliczności:
 - Użyj wywołania MQPUT, aby umieścić wiele komunikatów w kolejce *ta sama* .
Wywołanie MQOPEN z opcją MQOO_OUTPUT jest wysyłane jako pierwsze, po którym następuje jedna lub większa liczba żądań MQPUT w celu dodania komunikatów do kolejki. W końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1 .
 - Użyj wywołania MQPUT1 , aby umieścić tylko *jeden* komunikat w kolejce.
Wywołanie to enkapsuluje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wysłane.
2. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, kolejność tych komunikatów jest zachowywana, jeśli spełnione są określone warunki. Jednak w większości środowisk wywołanie MQPUT1 nie spełnia tych warunków, a więc nie zachowuje kolejności komunikatów. Zamiast tego w tych środowiskach należy użyć wywołania MQPUT. Szczegółowe informacje na ten temat zawiera sekcja [Uwagi dotyczące użycia MQPUT](#) .
3. Wywołania MQPUT1 mogą być używane do umieszczania komunikatów w listach dystrybucyjnych. Ogólne informacje na ten temat można znaleźć w uwagach dotyczących użycia dla wywołań MQOPEN i MQPUT.

Listy dystrybucyjne są obsługiwane w następujących środowiskach: klienci AIX, HP-UX, IBM i, Solaris, Linux, Windowsi IBM MQ połączone z tymi systemami.

W przypadku korzystania z wywołania MQPUT1 występują następujące różnice:

- a. Jeśli aplikacja udostępnia rekordy odpowiedzi MQRR, muszą one być udostępniane przy użyciu struktury MQOD. Nie można ich używać przy użyciu struktury MQPMO.
- b. Kod przyczyny MQRC_OPEN_FAILED nigdy nie jest zwracany przez MQPUT1 w rekordach odpowiedzi; jeśli kolejka nie zostanie otwarta, rekord odpowiedzi dla tej kolejki zawiera kod przyczyny wynikający z operacji otwarcia.

Jeśli operacja otwarcia dla kolejki powiedzie się z kodem zakończenia MQCC_WARNING, kod zakończenia i kod przyczyny w rekordzie odpowiedzi dla tej kolejki są zastępowane przez kody zakończenia i przyczyny wynikające z operacji put.

Podobnie jak w przypadku wywołań MQOPEN i MQPUT, menedżer kolejek ustawia rekordy odpowiedzi (jeśli są dostępne) tylko wtedy, gdy wynik wywołania nie jest taki sam dla wszystkich kolejek na liście dystrybucyjnej; jest to oznaczane przez wywołanie kończące się z kodem przyczyny MQRC_MULTIPLE_UZASADNIENIE.

4. Jeśli wywołanie MQPUT1 jest używane do umieszczania komunikatu w kolejce klastra, wywołanie zachowuje się tak, jakby w wywołaniu MQOPEN podano wartość MQOO_BIND_NOT_FIXED.
5. Jeśli na początku danych komunikatu aplikacji zostanie umieszczony komunikat zawierający jedną lub większą liczbę struktur nagłówka produktu IBM MQ, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka w celu sprawdzenia, czy są one poprawne. Więcej informacji na ten temat można znaleźć w uwagach dotyczących użycia wywołania MQPUT.
6. Jeśli wystąpi więcej niż jedna z sytuacji ostrzegawczych (patrz parametr *CompCode*), zwrócony kod przyczyny ma wartość *pierwsza* na następującej liście, która ma zastosowanie:
 - a. MQRC_MULTIPLE_POWODY
 - b. MQRC_INCOMPLETE_MSG
 - c. MQRC_INCOMPLETE_GROUP
 - d. MQRC_PRIORITY_PRZEKRACZA_MAKSIMUM lub MQRC_UNKNOWN_REPORT_OPTION
7. W przypadku języka programowania Visual Basic, zastosowanie mają następujące punkty:
 - Jeśli wielkość parametru *Buffer* jest mniejsza niż długość określona przez parametr *BufferLength*, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_BUFFER_LENGTH_ERROR.
 - Parametr *Buffer* jest zadeklarowany jako typ *String*. Jeśli dane, które mają być umieszczone w kolejce, nie są typu *String*, należy użyć Wywołanie MQPUT1Any w miejscu MQPUT1.Wywołanie MQPUT1Any ma takie same parametry, jak wywołanie MQPUT1, z tym wyjątkiem, że parametr *Buffer* jest zadeklarowany jako typ *Any*, co pozwala na umieszczanie w kolejce dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić *Buffer*, aby upewnić się, że wielkość ta wynosi co najmniej *BufferLength* bajtów.
8. Gdy wywołanie MQPUT1 jest wysyłane z MQPMO_SYNCPOINT, domyślne zachowanie zmienia się tak, że operacja put jest wykonywana asynchronicznie. Może to spowodować zmianę w zachowaniu niektórych aplikacji, które polegają na zwróconych określonych polach w strukturach MQOD i MQMD, ale które teraz zawierają niezdefiniowane wartości. Aplikacja może określić MQPMO_SYNC_RESPONSE, aby upewnić się, że operacja put jest wykonywana synchronicznie i że wszystkie odpowiednie wartości pól są zakończone.

Wywołanie C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,  
        BufferLength, Buffer, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */  
MQOD      ObjDesc;        /* Object descriptor */  
MQMD      MsgDesc;        /* Message descriptor */  
MQPMO     PutMsgOpts;     /* Options that control the action of MQPUT1 */  
MQLONG    BufferLength;    /* Length of the message in Buffer */  
MQBYTE    Buffer[n];       /* Message data */  
MQLONG    CompCode;       /* Completion code */  
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,  
                   BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN          PIC S9(9) BINARY.  
** Object descriptor  
01 OBJDESC.  
   COPY CMQODV.  
** Message descriptor  
01 MSGDESC.  
   COPY CMQMDV.  
** Options that control the action of MQPUT1  
01 PUTMSGOPTS.  
   COPY CMQPMOV.  
** Length of the message in BUFFER  
01 BUFFERLENGTH  PIC S9(9) BINARY.  
** Message data  
01 BUFFER         PIC X(n).  
** Completion code  
01 COMPCODE       PIC S9(9) BINARY.  
** Reason code qualifying COMPCODE  
01 REASON         PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,  
             CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn          fixed bin(31); /* Connection handle */  
dcl ObjDesc        like MQOD;    /* Object descriptor */  
dcl MsgDesc        like MQMD;    /* Message descriptor */  
dcl PutMsgOpts     like MQPMO;   /* Options that control the action of  
                                MQPUT1 */  
dcl BufferLength    fixed bin(31); /* Length of the message in Buffer */  
dcl Buffer          char(n);      /* Message data */  
dcl CompCode       fixed bin(31); /* Completion code */  
dcl Reason         fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X  
            BUFFER,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
OBJDESC	CMQODA	,	Object descriptor
MSGDESC	CMQMDA	,	Message descriptor
PUTMSGOPTS	CMQPMOA	,	Options that control the action of MQPUT1
BUFFERLENGTH	DS	F	Length of the message in BUFFER
BUFFER	DS	CL(n)	Message data
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,  
CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn      As Long   'Connection handle'  
Dim ObjDesc    As MQOD   'Object descriptor'  
Dim MsgDesc    As MQMD   'Message descriptor'  
Dim PutMsgOpts As MQPMO   'Options that control the action of MQPUT1'  
Dim BufferLength As Long   'Length of the message in Buffer'  
Dim Buffer      As String  'Message data'  
Dim CompCode   As Long   'Completion code'  
Dim Reason     As Long   'Reason code qualifying CompCode'
```

MQSET-ustawienie atrybutów obiektu

Użyj wywołania MQSET, aby zmienić atrybuty obiektu reprezentowanego przez uchwyt. Obiekt musi być kolejką.

Składnia

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Hobj

Typ: MQHOBJ-wejście

Ten uchwyt reprezentuje obiekt kolejki z atrybutami, które mają zostać ustawione. Uchwyt został zwrócony przez poprzednie wywołanie MQOPEN, które określiło opcję MQOO_SET.

SelectorCount

Typ: MQLONG-wejście

Jest to liczba selektorów, które są dostarczane w macierzy *Selectors* . Jest to liczba atrybutów, które mają zostać ustawione. Wartość zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

Selektory

Typ: MQLONGxSelectorCount-input

Jest to tablica selektorów atrybutów *SelectorCount* ; każdy selektor identyfikuje atrybut (liczba całkowita lub znak) z wartością, która ma zostać ustawiona.

Każdy selektor musi być poprawny dla typu kolejki reprezentowanej przez produkt *Hobj*. Dozwolone są tylko niektóre wartości MQIA_* i MQCA_*, które zostały wymienione w dalszej części listy.

Selektory mogą być określane w dowolnej kolejności. Wartości atrybutów, które odpowiadają selektorom atrybutów całkowitych (selektory MQIA_*), muszą być określone w *IntAttrs* w tej samej kolejności, w jakiej te selektory występują w produkcie *Selectors*. Wartości atrybutów, które odpowiadają selektorom atrybutów znakowych (selektory MQCA_*), muszą być określone w *CharAttrs* w tej samej kolejności, w jakiej występują te selektory. Selektory MQIA_* można przepląć z selektorami MQCA_*. Ważne jest tylko to, że kolejność względna w każdym typie jest istotna.

Ten sam selektor można określić więcej niż raz. Jeśli zostanie to określone, ostatnia wartość określona dla konkretnego selektora to ta, która staje się skuteczna.

Uwaga:

1. Selektory atrybutów całkowitoliczbowych i atrybutów znakowych są przydzielane w dwóch różnych zakresach; selektory MQIA_* znajdują się w zakresie MQIA_FIRST za pomocą MQIA_LAST, a selektory MQCA_* w zakresie MQCA_FIRST za pomocą MQCA_LAST.

Dla każdego zakresu wartości stałych MQIA_LAST_USED i MQCA_LAST_USED definiują najwyższą wartość akceptowania przez menedżer kolejek.

2. Jeśli wszystkie selektory MQIA_* występują jako pierwsze, te same numery elementów mogą być używane do adresowania odpowiednich elementów w macierzach *Selectors* i *IntAttrs*.
3. Jeśli parametr *SelectorCount* ma wartość zero, *Selectors* nie jest przywołany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

Atrybuty, które można ustawić, są wymienione w poniższej tabeli. Przy użyciu tego wywołania nie można ustawić żadnych innych atrybutów. W przypadku selektorów atrybutów MQCA_* stała, która definiuje długość w bajtach łańcucha, który jest wymagany w produkcie *CharAttrs*, jest podawana w nawiasach.

Tabela 110. Selektory atrybutów MQSET dla kolejek		
Selektor	Opis	Uwaga
MQCA_TRIGGER_DATA,	Dane wyzwalacza (MQ_TRIGGER_DATA_LENGTH).	
MQIA_DIST_LISTS	Obsługa listy dystrybucyjnej.	1
MQIA_INHIBIT_GET	Określa, czy operacje pobierania są dozwolone.	
MQIA_INHIBIT_PUT	Określa, czy operacje put są dozwolone.	
MQIA_TRIGGER_CONTROL	Sterowanie wyzwalaczem.	
MQIA_TRIGGER_DEPTH	Wyzwalacz uruchamiany zapełnieniem.	
MQIA_TRIGGER_MSG_PRIORITY,	Priorytet komunikatu progowego dla wyzwalaczy.	
MQIA_TRIGGER_TYPE	Typ wyzwalacza.	

Uwaga:

1. Obsługiwane tylko w systemach AIX, HP-UX, IBM i, Solaris, Linux, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.

IntAttrLiczba

Typ: MQLONG-wejście

Jest to liczba elementów w tablicy *IntAttrs* i musi być ona co najmniej liczba selektorów MQIA_* w parametrze *Selectors*. Wartość zero jest poprawną wartością, jeśli nie istnieje żadna wartość.

IntAttrs

Typ: MQLONGxIntAttrCount -wejście

Jest to tablica wartości atrybutu całkowitoliczbowego *IntAttrCount* . Te wartości atrybutów muszą być w tej samej kolejności, w jakiej znajdują się selektory MQIA_* w tablicy *Selectors* .

Jeśli parametr *IntAttrCount* lub *SelectorCount* ma wartość zero, *IntAttrs* nie jest przywołany; w tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CharAttrDługość

Typ: MQLONG-wejście

Jest to długość w bajtach parametru *CharAttrs* , która musi być co najmniej równa sumie długości atrybutów znakowych określonych w tablicy *Selectors* . Wartość zero jest poprawną wartością, jeśli w produkcie *Selectors* nie ma selektorów MQCA_*.

CharAttrs

Typ: MQCHARxCharAttrLength -wejście

Jest to bufor zawierający wartości atrybutów znakowych, które są konkatelowane. Długość buforu jest nadawana przez parametr *CharAttrLength* .

Atrybuty znaków muszą być określone w tej samej kolejności, w jakiej znajdują się selektory MQCA_* w tablicy *Selectors* . Długość każdego atrybutu znakowego jest stała (patrz *Selektory*). Jeśli wartość, która ma być ustawiona dla atrybutu, zawiera mniej niepustych znaków niż zdefiniowana długość atrybutu, należy dopełniać wartość w polu *CharAttrs* z prawej strony odstępami, aby wartość atrybutu była zgodna ze zdefiniowaną długością atrybutu.

Jeśli parametr *CharAttrLength* lub *SelectorCount* ma wartość zero, *CharAttrs* nie jest przywołany; w tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CF_NOT_AVAILABLE

(2345, X'929 ') Obiekt sprzęgający nie jest dostępny.

MQRC_CF_STRUC_NIE POWIODŁO SIĘ

(2373, X'945 ') Struktura CF (Coupling-Facility) nie powiodła się.

MQRC_CF_STRUC_IN_USE

(2346, X'92A') Struktura sprzęgania (Coupling-Facility) w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE

(2347, X'92B') Lista struktury narzędzia CF-nagłówków w użyciu.

MQRC_CHAR_ATTR_LENGTH_ERROR

(2006, X'7D6') Długość atrybutów znakowych nie jest poprawna.

MQRC_CHAR_ATTRS_ERROR

(2007, X'7D7') Łańcuch atrybutów znakowych nie jest poprawny.

MQRC_CICS_WAIT_FAILED (nie powiodło się)

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED

(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE

(2342, X'926 ') Podsystem Db2 nie jest dostępny.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_HOBJ_ERROR

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

Błąd MQRC_INHIBIT_VALUE_ERROR

(2020, X'7E4') Wartość atrybutu inhibit-get lub inhibit-put nie jest poprawna.

MQRC_INT_ATTR_COUNT_ERROR

(2021, X'7E5') Liczba atrybutów całkowitych nie jest poprawna.

MQRC_INT_ATTRS_ARRAY_ERROR

(2023, X'7E7') Tablica atrybutów Integer nie jest poprawna.

MQRC_NOT_OPEN_FOR_SET

(2040, X'7F8') Kolejka nie jest otwarta do ustawienia.

MQRC_OBJECT_CHANGED

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA

(2101, X'835 ') Obiekt jest uszkodzony.

BŁĄD MQRC_PAGESET_ERROR

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED

(2052, X'804 ') Kolejka została usunięta.

Błąd MQRC_Q_MGR_NAME_ERROR

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

MQRC_Q_MGR_NOT_AVAILABLE

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872 ') Menedżer kolejek jest zamykany.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SELECTOR_COUNT_ERROR,

(2065, X'811 ') Count of selectors not valid.

MQRC_SELECTOR_ERROR,

(2067, X'813 ') Selektor atrybutu nie jest poprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED

(2066, X'812 ') Count of selectors too large.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_SUPPRESSED_BY_EXIT

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

MQRC_TRIGGER_CONTROL_ERROR

(2075, X'81B') Wartość dla atrybutu sterującego wyzwalacza jest niepoprawna.

MQRC_TRIGGER_DEPTH_ERROR

(2076, X'81C') Wartość atrybutu głębokości wyzwalacza nie jest poprawna.

MQRC_TRIGGER_MSG_PRIORITY_ERR

(2077, X'81D') Wartość atrybutu wyzwalacza-message-priority nie jest poprawna.

MQRC_TRIGGER_TYPE_ERROR

(2078, X'81E') Wartość atrybutu wyzwalacza nie jest poprawna.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Za pomocą tego wywołania aplikacja może określić tablicę atrybutów całkowitoliczbowych lub kolekcję łańcuchów atrybutów znakowych. Jeśli nie wystąpią żadne błędy, podane atrybuty są ustawiane jednocześnie. W przypadku wystąpienia błędu (na przykład, jeśli selektor nie jest poprawny lub podjęto próbę ustawienia atrybutu na niepoprawną wartość), wywołanie nie powiedzie się i nie zostaną ustawione żadne atrybuty.
2. Wartości atrybutów można określić za pomocą wywołania MQINQ. Szczegółowe informacje można znaleźć w sekcji [“MQINQ-zapytanie o atrybuty obiektu”](#) na stronie 714 .
Uwaga: Nie wszystkie atrybuty z wartościami, które mogą być zapytane przy użyciu wywołania MQINQ, mogą mieć zmienione wartości przy użyciu wywołania MQSET. Na przykład w przypadku tego wywołania nie można ustawić atrybutów process-object lub queue-manager.
3. Zmiany atrybutów są zachowywane po restartach menedżera kolejek (inne niż zmiany w tymczasowych kolejkach dynamicznych, które nie są restartowane restartami menedżera kolejek).
4. Nie można zmieniać atrybutów kolejki modelowej przy użyciu wywołania MQSET. Jeśli jednak kolejka modelowa zostanie otwarta za pomocą wywołania MQOPEN z opcją MQOO_SET, można użyć wywołania MQSET w celu ustawienia atrybutów dynamicznej kolejki lokalnej utworzonej przy użyciu wywołania MQOPEN.
5. Jeśli ustawiony obiekt jest kolejką klastra, musi istnieć lokalna instancja kolejki klastra, aby możliwe było pomyślne wykonanie tej kolejki.

Więcej informacji na temat atrybutów obiektów zawiera sekcja:

- [“Atrybuty dla kolejek”](#) na stronie 845
- [“Atrybuty dla list nazw”](#) na stronie 879
- [“Atrybuty definicji procesów”](#) na stronie 881

- [“Atrybuty dla menedżera kolejek” na stronie 810](#)

Wywołanie C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
       CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ    Hobj;           /* Object handle */
MQLONG    SelectorCount;  /* Count of selectors */
MQLONG    Selectors[n];   /* Array of attribute selectors */
MQLONG    IntAttrCount;   /* Count of integer attributes */
MQLONG    IntAttrs[n];    /* Array of integer attributes */
MQLONG    CharAttrLength; /* Length of character attributes buffer */
MQCHAR    CharAttrs[n];   /* Character attributes */
MQLONG    CompCode;       /* Completion code */
MQLONG    Reason;         /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                  INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                  CHARATTRS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Object handle
01 HOBJ           PIC S9(9) BINARY.
** Count of selectors
01 SELECTORCOUNT PIC S9(9) BINARY.
** Array of attribute selectors
01 SELECTORS-TABLE.
02 SELECTORS      PIC S9(9) BINARY OCCURS n TIMES.
** Count of integer attributes
01 INTATTRCOUNT PIC S9(9) BINARY.
** Array of integer attributes
01 INTATTRS-TABLE.
02 INTATTRS       PIC S9(9) BINARY OCCURS n TIMES.
** Length of character attributes buffer
01 CHARATTRLENGTH PIC S9(9) BINARY.
** Character attributes
01 CHARATTRS      PIC X(n).
** Completion code
01 COMPCODE        PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON          PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
            IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn          fixed bin(31); /* Connection handle */
dcl Hobj           fixed bin(31); /* Object handle */
dcl SelectorCount  fixed bin(31); /* Count of selectors */
dcl Selectors(n)   fixed bin(31); /* Array of attribute selectors */
dcl IntAttrCount   fixed bin(31); /* Count of integer attributes */
dcl IntAttrs(n)    fixed bin(31); /* Array of integer attributes */
dcl CharAttrLength fixed bin(31); /* Length of character attributes
```

```

                                buffer */
dcl CharAttrs      char(n);      /* Character attributes */
dcl CompCode       fixed bin(31); /* Completion code */
dcl Reason         fixed bin(31); /* Reason code qualifying
                                CompCode */

```

Wywołanie High Level Assembler

```
CALL MQSET, (HCONN, HOBJ, SELECTORCOUNT, SELECTORS, INTATTRCOUNT, X
            INTATTRS, CHARATTRLENGTH, CHARATTRS, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HOBJ	DS	F	Object handle
SELECTORCOUNT	DS	F	Count of selectors
SELECTORS	DS	(n)F	Array of attribute selectors
INTATTRCOUNT	DS	F	Count of integer attributes
INTATTRS	DS	(n)F	Array of integer attributes
CHARATTRLENGTH	DS	F	Length of character attributes buffer
CHARATTRS	DS	CL(n)	Character attributes
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

Wywołanie języka Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

Dim Hconn	As Long	'Connection handle'
Dim Hobj	As Long	'Object handle'
Dim SelectorCount	As Long	'Count of selectors'
Dim Selectors	As Long	'Array of attribute selectors'
Dim IntAttrCount	As Long	'Count of integer attributes'
Dim IntAttrs	As Long	'Array of integer attributes'
Dim CharAttrLength	As Long	'Length of character attributes buffer'
Dim CharAttrs	As String	'Character attributes'
Dim CompCode	As Long	'Completion code'
Dim Reason	As Long	'Reason code qualifying CompCode'

MQSETMP-ustawienie właściwości komunikatu

Użyj wywołania MQSETMP, aby ustawić lub zmodyfikować właściwość uchwytu komunikatu.

Składnia

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytym połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*. Jeśli uchwyt komunikatu został utworzony przy użyciu komendy MQHC_UNASSOCIATED_HCONN, należy ustanowić poprawne połączenie w wątku ustawiające właściwość uchwytu komunikatu. W przeciwnym razie wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CONNECTION_BROKEN.

Hmsg

Typ: MQHMSG-wejście

To jest uchwyt komunikatu, który ma zostać zmodyfikowany. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

SetPropOpty

Typ: MQSMPO-wejście

Sterowanie sposobem ustawiania właściwości komunikatu.

Ta struktura umożliwia aplikacjom określanie opcji sterujących sposobem ustawiania właściwości komunikatu. Struktura jest parametrem wejściowym w wywołaniu MQSETMP. Więcej informacji na ten temat zawiera sekcja [MQSMPO](#).

nazwa

Typ: MQCHARV-wejście

Jest to nazwa właściwości do ustawienia.

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja [Nazwy właściwości i Ograniczenia dotyczące nazw właściwości](#).

PropDesc

Typ: MQPD-wejście/wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym:

- co się stanie, jeśli właściwość nie jest obsługiwana
- jaki kontekst komunikatu, do której należy właściwość
- Jakie komunikaty są kopiowane do postaci, w której jest ona kopiowana

Więcej informacji na temat tej struktury zawiera sekcja [MQPD](#).

typ

Typ: MQLONG-wejście

Typ danych dla ustawianej właściwości. Może to być jeden z następujących elementów:

MQTYPE_BOOLEAN

Wartość boolowska. *ValueLength* musi mieć wartość 4.

MQTYPE_BYTE_STRING

Łańcuch bajtów. *ValueLength* musi być równe zero lub większe.

MQTYPE_INT8

8-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 1.

MQTYPE_INT16

16-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 2.

MQTYPE_INT32

32-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 4.

MQTYPE_INT64

64-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 8.

MQTYPE_FLOAT32

32-bitowa liczba zmiennoprzecinkowa. *ValueLength* musi mieć wartość 4.

Uwaga: ten typ nie jest obsługiwany w przypadku aplikacji korzystających z produktu IBM w języku COBOL dla produktu z/OS.

MQTYPE_FLOAT64

64-bitową liczbę zmiennopozycyjną. *ValueLength* musi mieć wartość 8.

Uwaga: ten typ nie jest obsługiwany w przypadku aplikacji korzystających z produktu IBM w języku COBOL dla produktu z/OS.

MQTYPE_STRING

Łańcuch znaków. Wartość *ValueLength* musi być równa zero lub większa albo wartość specjalna MQVL_NULL_TERMINATED.

MQTYPE_NULL

Właściwość istnieje, ale ma wartość NULL. *ValueLength* musi mieć wartość zero.

ValueLength

Typ: MQLONG-wejście

Długość (w bajtach) wartości właściwości w parametrze *Wartość*. Wartość zero jest poprawna tylko dla wartości NULL lub łańcuchów lub łańcuchów bajtów. Wartość zero wskazuje, że właściwość istnieje, ale nie zawiera żadnych znaków ani bajtów.

Jeśli parametr *Type* ma ustawiony parametr MQTYPE_STRING, wartość ta musi być większa lub równa zero lub musi być równa zero lub być równa następującej wartości specjalnej:

MQVL_NULL_TERMINATED,

Wartość jest ograniczona do pierwszej wartości null napotkanej w łańcuchu. Wartość NULL nie jest uwzględniana jako część łańcucha. Ta wartość jest niepoprawna, jeśli parametr MQTYPE_STRING nie jest również ustawiony.

Uwaga: znak o kodzie zero używany do zakończenia łańcucha, jeśli parametr MQVL_NULL_TERMINATED jest ustawiony na wartość NULL, z zestawu znaków wartości.

wartość

Typ: MQBYTExValueDługość-wejście

Wartość właściwości, która ma zostać ustawiona. Bufor musi być wyrównany na granicy odpowiedniej do charakteru danych w wartości.

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void; adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *ValueLength* ma wartość zero, to nie jest przywołana wartość *Value*. W tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_WARNING:

Błąd formatu MQRC_RFH_FORMAT_ERROR

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'089C') Adapter nie jest dostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

MQRC_BUFFER_ERROR-BŁĄD

(2004, X'07D4') Parametr Wartość nie jest poprawny.

MQRC_BUFFER_LENGTH_ERROR

(2005, X'07D5') Parametr długości wartości nie jest poprawny.

MQRC_CALL_IN_PROGRESS

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_HMSG_ERROR

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

MQRC_MSG_HANDLE_IN_USE

(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

BŁĄD MQRC_PD_ERROR

(2482, X'09B2') Struktura deskryptora właściwości nie jest poprawna.

Błąd MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Niepoprawna nazwa właściwości.

MQRC_PROPERTY_TYPE_ERROR

(2473, X'09A9') Niepoprawny typ danych właściwości.

MQRC_PROP_NUMBER_FORMAT_ERROR

(2472, X'09A8') Napotkano błąd formatu liczb w danych wartości.

BŁĄD MQRC_SMPO_ERROR

(2463, X'099F') Ustawianie struktury opcji właściwości komunikatu nie jest poprawne.

MQRC_SOURCE_CCSID_ERROR, BŁĄD

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;          /* Connection handle */
MQHMSG    Hmsg;          /* Message handle */
MQSMPO    SetPropOpts;   /* Options that control the action of MQSETMP */
MQCHARV   Name;         /* Property name */
MQPD      PropDesc;      /* Property descriptor */
MQLONG    Type;         /* Property data type */
MQLONG    ValueLength;   /* Length of property value in Value */
MQBYTE    Value[n];      /* Property value */
MQLONG    CompCode;      /* Completion code */
MQLONG    Reason;        /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDSC, TYPE,  
                     VALUELENGTH, VALUE, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle  
01 HCONN      PIC S9(9) BINARY.  
** Message handle  
01 HMSG       PIC S9(18) BINARY.  
** Options that control the action of MQSETMP  
01 SETMSGOPTS.  
   COPY CMQSMPOV.  
** Property name  
01 NAME  
   COPY CMQCHRVV.  
** Property descriptor  
01 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.
```

Wywołanie PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,  
              Value, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */  
dcl Hmsg       fixed bin(63); /* Message handle */  
dcl SetPropOpts like MQSMPO; /* Options that control the action of MQSETMP */  
dcl Name       like MQCHARV; /* Property name */  
dcl PropDesc   like MQPD; /* Property descriptor */  
dcl Type       fixed bin(31); /* Property data type */  
dcl ValueLength fixed bin(31); /* Length of property value in Value */  
dcl Value      char(n); /* Property value */  
dcl CompCode   fixed bin(31); /* Completion code */  
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDSC,TYPE,VALUELENGTH,  
              VALUE,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
HMSG	DS	D	Message handle
SETMSGOPTS	CMQSMPOA	,	Options that control the action of MQSETMP
NAME	CMQCHRVV	,	Property name
PROPDSC	CMQPDV	,	Property descriptor
TYPE	DS	F	Property data type
VALUELENGTH	DS	F	Length of property value in VALUE
VALUE	DS	CL(n)	Property value

COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSTAT-pobieranie informacji o statusie

Użyj wywołania MQSTAT, aby pobrać informacje o statusie. Typ zwracanych informacji o statusie jest określany na podstawie wartości typu określonej w wywołaniu.

Składnia

MQSTAT (*Hconn*, *Type*, *Stat*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilewanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

typ

Typ: MQLONG-wejście

Typ żądanych informacji o statusie. Poprawne wartości to:

MQSTAT_TYPE_ASYNC_ERROR,

Zwraca informacje na temat poprzednich asynchronicznych operacji put.

MQSTAT_TYPE_RECONNECTION

Zwraca informacje o ponownym połączeniu. Jeśli połączenie nawiąże ponownie połączenie lub nie nawiąże ponownie połączenia, informacje te opisują niepowodzenie, które spowodowało ponowne nawiązanie połączenia.

Ta wartość jest poprawna tylko dla połączeń klientów. W przypadku innych typów połączeń wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_ENVIRONMENT_ERROR**

MQSTAT_TYPE_RECONNECTION_ERROR

Zwracane są informacje o poprzedniej awarii związanej z ponownym nawiązaniem połączenia. Jeśli ponowne nawiązanie połączenia nie powiodło się, informacje te opisują niepowodzenie, które spowodowało niepowodzenie ponownego nawiązania połączenia.

Ta wartość jest poprawna tylko dla połączeń klientów. W przypadku innych typów połączeń wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_ENVIRONMENT_ERROR**.

stat (stat)

Typ: MQSTS-input/output

Struktura informacji o statusie. Szczegółowe informacje na ten temat zawiera sekcja [“MQSTS-struktura raportowania statusu”](#) na stronie 595.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_API_EXIT_ERROR

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się

MQRC_API_EXIT_LOAD_ERROR

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_CALL_IN_PROGRESS

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

MQRC_CONNECTION_BROKEN

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_ZATRZYMYWANIE

(2203, X'89B') Połączenie jest zamykane.

MQRC_FUNCTION_NOT_SUPPORTED

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

MQRC_Q_MGR_ZATRZYMYWANIE

(2162, X'872')-Zatrzymywanie menedżera kolejek

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

Błąd MQRC_STAT_TYPE_ERROR

(2430, X'97E' Błąd typu MQSTAT

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

BŁĄD MQRC_STS_ERROR

(2426, X'97A') Błąd struktury MQSTS

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

1. Wywołanie funkcji MQSTAT określające typ MQSTAT_TYPE_ASYNC_ERROR zwraca informacje o poprzednich operacjach asynchronicznych MQPUT i MQPUT1 . Struktura MQSTS przekazana z powrotem po powrocie z wywołania MQSTAT zawiera pierwsze zarejestrowane asynchroniczne ostrzeżenie lub informacje o błędach dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia będą następowały po pierwszym, nie zmieniają one zwykle tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia MQCC_WARNING, to zamiast tego zwracana jest następna awaria z kodem zakończenia MQCC_FAILED .

2. Jeśli od czasu nawiązania połączenia nie wystąpiły żadne błędy lub od ostatniego wywołania do MQSTAT , w strukturze MQSTS zwracane są CompCode z MQCC_OK i przyczyna MQRC_NONE .
3. Liczby wywołań asynchronicznych, które zostały przetworzone w ramach uchwytu połączenia, są zwracane w postaci trzech pól licznika: PutSuccessCount, PutWarningCount i PutFailureCount. Liczniki te są zwiększane przez menedżer kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy zwrócić uwagę, że w celach księgowych lista dystrybucyjna jest liczona raz dla kolejki docelowej, a nie raz na listę dystrybucyjną). Licznik nie jest zwiększany po przekroczeniu maksymalnej wartości dodatniej AMQ_LONG_MAX.
4. Pomyślne wywołanie programu MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczby.
5. Zachowanie produktu MQSTAT zależy od wartości parametru MQSTAT Type , który jest podany.
6. **MQSTAT_TYPE_ASYNC_ERROR,**
 - a. Wywołanie funkcji MQSTAT określające typ MQSTAT_TYPE_ASYNC_ERROR zwraca informacje o poprzednich operacjach asynchronicznych MQPUT i MQPUT1 . Struktura MQSTS przekazana z powrotem po powrocie z wywołania MQSTAT zawiera pierwsze zarejestrowane asynchroniczne ostrzeżenie lub informacje o błędach dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia będą następowały po pierwszym, nie zmieniają one zwykle tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia MQCC_WARNING, to zamiast tego zwracana jest następna awaria z kodem zakończenia MQCC_FAILED .
 - b. Jeśli od czasu nawiązania połączenia nie wystąpiły żadne błędy lub od ostatniego wywołania do MQSTAT , w strukturze MQSTS zwracane są CompCode z MQCC_OK i przyczyna MQRC_NONE .
 - c. Liczby wywołań asynchronicznych, które zostały przetworzone w ramach uchwytu połączenia, są zwracane w postaci trzech pól licznika: PutSuccessCount, PutWarningCount i PutFailureCount. Liczniki te są zwiększane przez menedżer kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy zwrócić uwagę, że w celach księgowych lista dystrybucyjna jest liczona raz dla kolejki docelowej, a nie raz na listę dystrybucyjną). Licznik nie jest zwiększany po przekroczeniu maksymalnej wartości dodatniej AMQ_LONG_MAX.
 - d. Pomyślne wywołanie programu MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczby.

MQSTAT_TYPE_RECONNECTION

Przypuśćmy, że w trakcie ponownego nawiązywania połączenia wywoływana jest MQSTAT z Type ustawionym na MQSTAT_TYPE_RECONNECTION wewnątrz procedury obsługi zdarzeń. Należy rozważyć poniższe przykłady.

Klient podejmuje próbę ponownego nawiązania połączenia lub nie udało się nawiązać połączenia.

CompCode w strukturze MQSTS to MQCC_FAILED , a Reason może mieć wartość MQRC_CONNECTION_BROKEN lub MQRC_Q_MGR QUIESCING. ObjectType is MQOT_Q_MGR, ObjectName is the name of the queue manager, and ObjectQMGrName is blank.

Klient pomyślnie nawiąże ponowne połączenie lub nigdy nie został odłączony.

CompCode w strukturze MQSTS to MQCC_OK , a Reason to MQRC_NONE

Kolejne wywołania programu MQSTAT zwracają te same wyniki.

MQSTAT_TYPE_RECONNECTION_ERROR

Przypuśćmy, że wywoła MQSTAT z Type ustawionym na MQSTAT_TYPE_RECONNECTION_ERROR w odpowiedzi na odezwę MQRC_RECONNECT_FAILED do wywołania MQI. Należy rozważyć poniższe przykłady.

Niepowodzenie autoryzacji podczas ponownego otwarcia kolejki podczas ponownego nawiązania połączenia z innym menedżerem kolejek.

CompCode w strukturze MQSTS to MQCC_FAILED , a Reason jest przyczyną niepowodzenia ponownego nawiązania połączenia, takiego jak MQRC_NOT_AUTHORIZED. ObjectType to typ

obiektu, który spowodował problem, taki jak MQOT_QUEUE, ObjectName to nazwa kolejki, a ObjectQMgrName nazwa menedżera kolejek będącego właścicielem kolejki.

Podczas ponownego nawiązania połączenia wystąpił błąd połączenia gniazda.

CompCode w strukturze MQSTS to MQCC_FAILED, a Reason jest przyczyną niepowodzenia ponownego nawiązania połączenia, takiego jak MQRC_HOST_NOT_AVAILABLE. ObjectType is MQOT_Q_MGR, ObjectName is the name of the queue manager, and ObjectQMgrName is blank.

Kolejne wywołania programu MQSTAT zwracają te same wyniki.

Wywołanie C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;          /* Connection Handle */
MQLONG StatType;        /* Status type */
MQSTS Stat;             /* Status information structure */
MQLONG CompCode;        /* Completion code */
MQLONG Reason;          /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**      Connection handle
01      HCONN      PIC S9(9)          BINARY.
**      Status type
01      STATTYPE   PIC S9(9)          BINARY.
**      Status information
01      STAT.
      COPY CMQSTSV.
**      Completion code
01      COMPCODE   PIC S9(9)          BINARY.
**      Reason code qualifying COMPCODE
01      REASON     PIC S9(9)          BINARY.
```

Wywołanie PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn      fixed bin(31); /* Connection handle */
dcl StatType   fixed bin(31); /* Status type */
dcl Stat       like MQSTS;    /* Status information structure */
dcl CompCode   fixed bin(31); /* Completion code */
dcl Reason     fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie asemblera System/390

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
STATTYPE	DS	F	Status type
STAT	CMQSTSA,		Status information structure
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQSUB-Zarejestruj subskrypcję

Użyj wywołania MQSUB, aby zarejestrować subskrypcję aplikacji do konkretnego tematu.

Składnia

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Przyczyna*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

SubDesc

Typ: MQSD-input/output

Jest to struktura identyfikująca używany obiekt, który jest rejestrowany przez aplikację. Więcej informacji na ten temat zawiera sekcja [“MQSD-deskryptor subskrypcji”](#) na stronie 568 .

Hobj

Typ: MQHOBJ-input/output

Ten uchwyt reprezentuje dostęp, który został utworzony w celu uzyskania komunikatów wysłanych do tej subskrypcji. Komunikaty te mogą być przechowywane w określonej kolejce lub menedżer kolejek może zarządzać pamięcią masową bez użycia określonej kolejki.

Aby korzystać z określonej kolejki, należy ją powiązać z subskrypcją podczas tworzenia subskrypcji. Można to zrobić na dwa sposoby:

- Używając komendy DEFINE SUB MQSC, i pod warunkiem, że komenda ta ma nazwę obiektu kolejki.
- Udostępniając ten uchwyt podczas wywoływania funkcji MQSUB za pomocą komendy MQSO_CREATE

Jeśli ten uchwyt jest podany jako parametr wejściowy w wywołaniu, musi to być poprawny uchwyt obiektu zwrócony z poprzedniego wywołania MQOPEN w kolejce przy użyciu co najmniej jednej z następujących opcji:

- MQOO_INPUT_*
- MQOO_BROWSE
- MQOO_OUTPUT (jeśli kolejka jest kolejką zdalną)

W przeciwnym razie wywołanie zakończy się niepowodzeniem z błędem MQRC_HOBJ_ERROR. Nie może to być uchwyt obiektu do kolejki aliasowej, która jest tłumaczona na obiekt tematu. Jeśli tak, wywołanie zakończy się niepowodzeniem z błędem MQRC_HOBJ_ERROR.

Jeśli menedżer kolejek ma zarządzać pamięcią masową komunikatów wysłanych do tej subskrypcji, należy ustawić tę opcję podczas tworzenia subskrypcji, używając opcji MQSO_MANAGED. Następnie menedżer kolejek zwraca ten uchwyt jako parametr wyjściowy w wywołaniu. Zwracany uchwyt jest znany jako uchwyt zarządzany. Jeśli określono parametr MQHO_NONE, ale nie określono MQSO_MANAGED, wywołanie zakończy się niepowodzeniem z błędem MQRC_HOBJ_ERROR.

Gdy menedżer kolejek zwrócił do użytkownika uchwyt zarządzany, można go użyć w wywołaniu MQGET lub MQCB z opcjami przeglądania lub bez, w wywołaniu MQINQ lub w tabeli MQCLOSE. Nie można jej użyć w przypadku operacji MQPUT, MQSUB, MQSET; próba wykonania tego działania kończy się niepowodzeniem z parametrem MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR lub MQRC_NOT_OPEN_FOR_SET.

Jeśli ta subskrypcja jest wznawiana za pomocą opcji MQSO_RESUME w strukturze MQSD, uchwyt może zostać zwrócony do aplikacji w tym parametrze, ustawiając parametr MQSO_MANAGED na wartość MQHO_NONE. Można to zrobić, niezależnie od tego, czy subskrypcja używa zarządzanego uchwyty, czy też nie, i może być przydatne udostępnianie subskrypcji utworzonych przy użyciu opcji DEFINE SUB z uchwytami do kolejki subskrypcji zdefiniowanej w tej komendzie. W przypadku, gdy wznawiana jest administracyjna subskrypcja, zostaje otwarta kolejka z opcją MQOO_INPUT_AS_Q_DEF i MQOO_BROWSE. Jeśli konieczne jest określenie innych opcji, aplikacja musi jawnie otworzyć kolejkę subskrypcji i udostępnić uchwyt obiektu w wywołaniu. Jeśli wystąpi problem z otwarciem kolejki, wywołanie komendy MQRC_INVALID_DESTINATION nie powiedzie się. Jeśli zostanie podana *Hobj*, musi to być odpowiednik *Hobj* w oryginalnym wywołaniu MQSUB. Oznacza to, że jeśli udostępniany jest uchwyt obiektu zwrócony z wywołania MQOPEN, uchwyt musi znajdować się w tej samej kolejce, co poprzednio używane. Jeśli nie jest to ta sama kolejka, wywołanie kończy się niepowodzeniem z błędem MQRC_HOBJ_ERROR.

Jeśli ta subskrypcja jest zmieniana za pomocą opcji MQSO_ALTER w strukturze MQSD, można podać inną wartość *Hobj*. Wszystkie publikacje, które zostały dostarczone do kolejki i które zostały wcześniej zidentyfikowane za pomocą tego parametru, pozostają w tej kolejce i za pomocą aplikacji należy pobrać te komunikaty, jeśli parametr *Hobj* reprezentuje teraz inną kolejkę.

W tabeli podsumowano użycie tego parametru z różnymi opcjami subskrypcji:

Opcje	<i>Hobj</i>	Opis
MQSO_CREATE + MQSO_MANAGED	Ignorowane na wejściu	Tworzy subskrypcję przy użyciu pamięci masowej komunikatów zarządzanych przez menedżera kolejek.
MQSO_CREATE	Poprawny uchwyt obiektu	Tworzy subskrypcję udostępniając określoną kolejkę jako miejsce docelowe dla komunikatów.
MQSO_RESUME	MQHO_NONE	Wznawia wcześniej utworzoną subskrypcję, niezależnie od tego, czy była ona zarządzana, czy też nie, a menedżer kolejek zwraca uchwyt obiektu do użycia przez aplikację.
MQSO_RESUME	Poprawny, zgodny uchwyt obiektu	Wznawia wcześniej utworzoną subskrypcję, która używa określonej kolejki jako miejsca docelowego dla komunikatów i korzysta z uchwyty obiektu z określonymi opcjami otwierania.

Opcje	Hobj	Opis
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Zmienia istniejącą subskrypcję, która wcześniej była używana w określonej kolejce, więc jest to subskrypcja zarządzana. Nie można zmienić klasy miejsca docelowego (zarządzanego lub nie).
MQSO_ALTER	Poprawny uchwyt obiektu	Zmienia istniejącą subskrypcję, niezależnie od tego, czy była ona zarządzana, czy nie, tak aby teraz używała konkretnej kolejki. Jeśli opcja MQSO_MANAGED nie jest używana, udostępniona kolejka może zostać zmieniona, ale klasa miejsca docelowego (zarządzana lub nie) nie może zostać zmieniona.

W kolejnych wywołaniach MQGET lub MQCB, *Hobj* mają odbierać komunikaty publikowania wysyłane do tej subskrypcji, należy określić, czy został on udostępniony, czy zwrócony.

Uchwyt *Hobj* nie jest już poprawny, gdy na nim zostanie wywołana wywołanie MQCLOSE lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostaje zakończona (do momentu rozłączenia aplikacji). Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Informacje na temat zasięgu uchwytu znajdują się w sekcji [Hconn \(MQHCONN\)-dane wyjściowe](#) . Operacja MQCLOSE uchwytu *Hobj* nie ma wpływu na uchwyt *Hsub* .

Hsub

Typ: MQHOBJ-wyjście

Ten uchwyt reprezentuje subskrypcję, która została wykonana. Może być używany do dwóch kolejnych operacji:

- Można jej użyć w kolejnym wywołaniu MQSUBRQ, aby zażądać wystania publikacji, gdy opcja MQSO_PUBLICATIONS_ON_REQUEST została użyta podczas dokonywania subskrypcji.
- Można go użyć podczas kolejnego wywołania MQCLOSE w celu usunięcia subskrypcji, która została wykonana. Uchwyt *Hsub* przestaje być poprawny, gdy zostanie wywołana wywołanie MQCLOSE lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostanie zakończona. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Operacja MQCLOSE uchwytu *Hsub* nie ma wpływu na uchwyt *Hobj* .

Ten uchwyt nie może zostać przekazany do wywołania MQGET lub MQCB. Należy użyć parametru *Hobj* . Tego uchwytu nie można używać w żadnym wywołaniu programu IBM MQ innym niż MQCLOSE lub MQSUBRQ. Przekazanie tego uchwytu do dowolnego innego wywołania programu IBM MQ spowoduje wywołanie MQRC_HOBJ_ERROR.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Pomyślne zakończenie

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK, kod przyczyny jest następujący:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED, kodem przyczyny jest jeden z następujących kodów:

MQRC_CLUSTER_RESOLUTION_ERROR

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

MQRC_DURABILITY_NOT_ALLOWED

2436 (X'0984 ') Wywołanie MQSUB przy użyciu opcji MQSO_DURABLE nie powiodło się.

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

BŁĄD MQRC_HOBJ_ERROR

2019 (X'07E3') Uchwyt obiektu Hobj nie jest poprawny.

Niezgodność MQRC_IDENTITY_MISMATCH

2434 (X'0982 ') Nazwa subskrypcji jest zgodna z istniejącą subskrypcją.

MQRC_NOT_AUTHORIZED

2035 (X'07F3') Użytkownik nie ma uprawnień do wykonania operacji.

MQRC_NO_SUBSCRIPTION

2428 (X'097C') zidentyfikowana nazwa subskrypcji nie istnieje.

MQRC_OBJECT_STRING_ERROR,

2441 (X'0989 ') Pole Objectstring nie jest poprawne.

BŁĄD MQRC_OPTIONS_ERROR

2046 (X'07FE') Parametr lub pole opcji zawiera opcje, które są niepoprawne, lub kombinacja opcji, która jest niepoprawna.

MQRC_Q_MGR QUIESCING,

2161 (X'0871 ') Menedżer kolejek wygaszany.

MQRC_RECONNECT_Q_MGR_REQD

2555 (X'09FB' X) Wymagana jest opcja MQCNO_RECONNECT_Q_MGR.

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać pobrane.

MQRC_RETAINED_NOT_DOSTARCZONEGO

2526 (X'09DE') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać dostarczone do kolejki docelowej subskrypcji i nie mogą zostać dostarczone do kolejki niedostarczonych komunikatów.

Błąd MQRC_SD_ERROR

2424 (X'0978 ') deskryptor subskrypcji (MQSD) jest niepoprawny.

MQRC_SELECTION_NOT_AVAILABLE

2551 (X'09F7') Łańcuch wyboru nie jest zgodny ze składnią selektora IBM MQ i nie był dostępny żaden dostawca wyboru rozszerzonego komunikatu.

Błąd MQRC_SELECTION_STRING_ERROR

2519 (X'09D7') Łańcuch wyboru musi zostać określony zgodnie z opisem w dokumentacji struktury MQCHARV.

MQRC_SELECTOR_SYNTAX_ERROR

2459 (X'099B') Wywołanie MQOPEN, MQPUT1 lub MQSUB zostało wydane, ale podano łańcuch wyboru, który zawierał błąd składniowy.

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') Pole danych SubUser nie jest poprawne.

MQRC_SUB_NAME_ERROR-BŁĄD

2440 (X'0988 ') Pole SubName jest niepoprawne.

MQRC_SUB_ALREADY_EXISTS

2432 (X'0980 ') Subskrypcja już istnieje.

MQRC_SUB_USER_DATA_ERROR

2431 (X'097F') Pole danych SubUsernie jest poprawne.

Błąd MQRC_TOPIC_STRING_ERROR

2425 (X'0979 ') Łańcuch tematu nie jest poprawny.

MQRC_UNKNOWN_OBJECT_NAME

2085 (X'0825 ') Nie można znaleźć obiektu zidentyfikowanego w polu MQSD ObjectName .

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

- Subskrypcja jest tworzona w temacie o nazwie określonej za pomocą skróconej nazwy predefiniowanego obiektu tematu, pełnej nazwy łańcucha tematu lub jest tworzona przez konkatencję dwóch części. Patrz opis produktów *ObjectName* i *ObjectString* w podręczniku [“MQSD-deskryptor subskrypcji”](#) na stronie 568.
 - Menedżer kolejek wykonuje sprawdzenia zabezpieczeń po wywołaniu wywołania MQSUB w celu sprawdzenia, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma odpowiedni poziom uprawnień, zanim dostęp jest dozwolony. Odpowiedni obiekt tematu znajduje się w hierarchii tematów, a na tym obiekcie tematu jest sprawdzany uprawnienia, aby upewnić się, że ustawione są uprawnienia do subskrybowania. Jeśli opcja MQSO_MANAGED nie jest używana, w kolejce docelowej wykonywane jest sprawdzenie uprawnień, aby upewnić się, że ustawione są uprawnienia dla danych wyjściowych. Jeśli używana jest opcja MQSO_MANAGED, nie jest wykonywane sprawdzanie uprawnień do kolejki zarządzanej w celu uzyskania dostępu lub uzyskania dostępu do zapytania.
 - Jeśli jako dane wejściowe nie zostanie dostarczona usługa Hobj, wywołanie MQSUB przydziela dwa uchwyty, uchwyt obiektu (Hobj) i uchwyt subskrypcji (Hsub).
 - Jeśli używana jest opcja MQSO_MANAGED, w wywołaniu MQSUB zwracany jest identyfikator Hobj, który można sprawdzić w celu znalezienia atrybutów, takich jak próg wycofania i nazwa nadmiernej kolejki wycofanych komunikatów. Można również zapytać o nazwę kolejki zarządzanej, ale nie można próbować bezpośrednio otwierać tej kolejki.
 - Subskrypcje można grupować, zezwalając na dostarczanie tylko jednej publikacji do grupy subskrypcji, nawet jeśli więcej niż jedna grupa jest zgodna z publikacją. Subskrypcje są grupowane za pomocą opcji MQSO_GROUP_SUB, a także w celu grupowania subskrypcji, które muszą być
 - korzystanie z tej samej kolejki nazwanej (która nie używa opcji MQSO_MANAGED) w tym samym menedżerze kolejek-reprezentowany przez parametr Hobj w wywołaniu MQSUB
 - współużytkuj ten sam identyfikator SubCorrel
 - być z tego samego SubLevel
- Atrybuty te definiują zestaw subskrypcji uważanych za znajdujące się w grupie, a także atrybuty, których nie można zmienić, jeśli subskrypcja została pogrupowana. Zmiana wartości SubLevel w tabeli MQRC_SUBLEVEL_NOT_ALTERABLE oraz zmiana dowolnego z pozostałych (które mogą zostać zmienione, jeśli subskrypcja nie jest zgrupowana) powoduje, że MQRC_GROUPING_NOT_ALTERABLE nie jest możliwe.
- Pomyślne zakończenie wywołania MQSUB nie oznacza, że działanie zostało zakończone. Aby sprawdzić, czy to wywołanie zostało zakończone, należy przejść do kroku DEFINE SUB (DEFINE SUB) w sekcji [Sprawdzanie, czy komendy asynchroniczne dla sieci rozproszonych zostały zakończone](#).
 - Pola w tabeli MQSD są wypełniane w odpowiedzi na zwrot z wywołania MQSUB, który korzysta z opcji MQSO_RESUME. Zwracaną wartość MQSD można przekazać bezpośrednio do wywołania MQSUB, które korzysta z opcji MQSO_ALTER z dowolnymi zmianami, które należy wprowadzić w subskrypcji zastosowanej w MQSD. Niektóre pola mają specjalne uwagi, które zostały odnotowane w tabeli.

Dane wyjściowe MQSD z tabeli MQSUB	
Nazwa pola w MQSD	Uwagi szczególne
Opcje dostępu lub tworzenia	Niektóre opcje mogą być resetowane po powrocie z wywołania MQSUB. Jeśli zmaterializowana tabela MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB, wymagana opcja musi być jawnie ustawiona.
Opcje trwałości, Opcje docelowe, Opcje rejestracji i opcje wieloznaczne	Te opcje są ustawione jako odpowiednie
Opcje publikacji	Te opcje są ustawione jako odpowiednie, z wyjątkiem MQSO_NEW_PUBLICATIONS_ONLY, które mają zastosowanie tylko do MQSO_CREATE.
Inne opcje	Te opcje nie zmieniają się po powrocie z wywołania MQSUB. Sterują one sposobem, w jaki wywołanie API jest wysyłane i nie jest zapisywane w subskrypcji. Muszą one być ustawione zgodnie z wymaganiami w przypadku kolejnych wywołań MQSUB ponownie korzystających z MQSD.
ObjectName	To pole wejściowe jest tylko niezmienione w przypadku zwrotu z wywołania MQSUB.
ObjectString	To pole wejściowe jest tylko niezmienione w przypadku zwrotu z wywołania MQSUB. W polu <i>ResObjectString</i> zwracana jest pełna nazwa tematu, jeśli został podany bufor.
Identyfikator AlternateUser i identyfikator AlternateSecurity	Te pola wejściowe są tylko niezmienione po powrocie z wywołania MQSUB. Sterują one sposobem, w jaki wywołanie API jest wysyłane i nie jest zapisywane w subskrypcji. Muszą one być ustawione zgodnie z wymaganiami w przypadku kolejnych wywołań MQSUB ponownie korzystających z wywołania MQSD.
SubExpiry	W przypadku powrotu z wywołania MQSUB przy użyciu opcji MQSO_RESUME to pole jest ustawione na pierwotny limit czasu utraty ważności subskrypcji, a nie pozostały czas utraty ważności. Jeśli zmaterializowana tabela MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB za pomocą opcji MQSO_ALTER, należy zresetować limit czasu utraty ważności subskrypcji, aby ponownie rozpocząć zliczanie czasu.
SubName	To pole jest polem wejściowym w wywołaniu MQSUB i nie jest zmieniane na wyjściu.

Dane wyjściowe MQSD z tabeli MQSUB (kontynuacja)	
Nazwa pola w MQSD	Uwagi szczególne
Dane SubUseri SelectionString	Te pola długości zmiennej są zwracane w wyniku wywołania MQSUB przy użyciu opcji MQSO_RESUME, jeśli bufor jest udostępniany, a także dodatniej długości buforu w produkcie <i>VSBufSize</i>. Jeśli bufor nie zostanie podany, tylko długość jest zwracana w polu <i>VSLength</i> komendy MQCHARV. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko <i>VSBufSize</i> bajtów. Jeśli zmaterializowana tabela MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB za pomocą opcji MQSO_ALTER, a bufor nie zostanie podany, ale zostanie podana wartość inna niż zero <i>VSLength</i>, to jeśli ta długość jest zgodna z istniejącą długością pola, nie zostanie wykonana żadna zmiana w tym polu.
Identyfikator SubCorreli znacznik PubAccounting	Jeśli nie zostanie użyty parametr MQSO_SET_CORREL_ID, <i>SubCorrelId</i> zostanie wygenerowany przez menedżer kolejek. Jeśli nie jest używany parametr MQSO_SET_IDENTITY_CONTEXT, to <i>PubAccountingToken</i> jest generowany przez menedżer kolejek. Te pola są zwracane w tabeli MQSD z wywołania MQSUB przy użyciu opcji MQSO_RESUME. Jeśli są one generowane przez menedżer kolejek, wygenerowana wartość jest zwracana w wywołaniu MQSUB za pomocą opcji MQSO_CREATE lub MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Te pola są zwracane w tabeli MQSD.
Łańcuch ResObject	To pole wyjściowe jest zwracane w zmaterializowanych tabelach zapytań (MQSD), jeśli podano bufor.

Wywołanie C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;    /* Connection handle */
MQSD SubDesc;     /* Subscription descriptor */
MQHOBJ Hobj;      /* Object handle */
MQHOBJ Hsub;      /* Subscription handle */
MQLONG CompCode;  /* Completion code */
MQLONG Reason;    /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN    PIC S9(9) BINARY.
** Subscription descriptor
01 SUBDESC.
   COPY CMQSDV.
** Object handle
01 HOBJ     PIC S9(9) BINARY.
** Subscription handle
01 HSUB     PIC S9(9) BINARY.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON   PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn    fixed bin(31); /* Connection handle */
dcl SubDesc  like MQSD;    /* Subscription descriptor */
dcl Hobj     fixed bin(31); /* Object handle */
dcl Hsub     fixed bin(31); /* Subscription handle */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason   fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQSUB, (HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN    DS      F  Connection handle
SUBDESC  CMQSDA  ,  Subscription descriptor
HOBJ     DS      F  Object handle
HSUB     DS      F  Subscription handle
COMPCODE DS      F  Completion code
REASON   DS      F  Reason code qualifying COMPCODE
```

MQSUBRQ-żądanie subskrypcji

Należy użyć wywołania MQSUBRQ w celu złożenia żądania dla zachowanej publikacji, gdy subskrybent został zarejestrowany w tabeli MQSO_PUBLICATIONS_ON_REQUEST.

Składnia

MQSUBRQ (*Hconn*, *Hsub*, *Action*, *SubRqOpts*, *Compcode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *Hconn* :

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Hsub

Typ: MQHOBJ-wejście

Ten uchwyt reprezentuje subskrypcję, dla której ma zostać zamówiona aktualizacja. Wartość *Hsub* została zwrócona z poprzedniego wywołania MQSUB.

Action

Typ: MQLONG-wejście

Ten parametr steruje konkretnymi działaniami, które są żądane w subskrypcji. Należy podać następującą wartość:

MQSR_ACTION_PUBLICATION

To działanie wymaga, aby publikacja aktualizacji została wysłana dla określonego tematu. Można go używać tylko wtedy, gdy subskrybent określił opcję MQSO_PUBLICATIONS_ON_REQUEST w wywołaniu MQSUB, gdy została ona wykonana. Jeśli menedżer kolejek ma zachowaną publikację dla tematu, jest ona wysyłana do subskrybenta. Jeśli nie, wywołanie nie powiedzie się. Jeśli aplikacja jest wysyłana do publikacji, która została zachowana, jest ona wskazana przez właściwość komunikatu MQIsRetained w tej publikacji.

Ponieważ temat w istniejącej subskrypcji reprezentowanej przez parametr *Hsub* może zawierać znaki wieloznaczne, subskrybent może otrzymać wiele zachowanych publikacji.

SubRqOpty

Typ: MQSRO-input/output

Te opcje sterują działaniem MQSUBRQ, patrz [“MQSRO-opcje żądania subskrypcji” na stronie 592](#), aby uzyskać szczegółowe informacje.

Jeśli żadne opcje nie są wymagane, programy napisane w języku C lub S/390 assembler mogą określać pusty adres parametru zamiast określać adres struktury MQSRO.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK

Pomyślne zakończenie

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED

2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_NO_RETAINED_MSG,

2437 (X'0985 ') Nie ma zachowanych publikacji obecnie przechowywanych dla tego tematu.

BŁĄD MQRC_OPTIONS_ERROR

2046 (X'07FE') Parametr lub pole opcji zawiera opcje, które są niepoprawne, lub kombinacja opcji, która jest niepoprawna.

MQRC_Q_MGR QUIESCING,

2161 (X'0871 ') Menedżer kolejek wygaszany.

BŁĄD MQRC_SRO_ERROR

2438 (X'0986 ') W wywołaniu MQSUBRQ opcje żądania subskrypcji MQSRO nie są poprawne.

MQRC_RETAINED_MSG_Q_ERROR

2525 (X'09DD') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać pobrane.

MQRC_RETAINED_NOT_DOSTARCZONEGO

2526 (X'09DE') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać dostarczone do kolejki docelowej subskrypcji i nie mogą zostać dostarczone do kolejki niedostarczonych komunikatów.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Użycie notatek

Następujące uwagi dotyczące użycia mają zastosowanie do użycia kodu działania

MQSR_ACTION_PUBLICATION:

1. Jeśli komenda zakończy się pomyślnie, zachowane publikacje zgodne z podaną subskrypcją zostały wysłane do subskrypcji i można je odebrać za pomocą komendy MQGET lub MQCB, korzystając z komendy Hobj zwróconej w oryginalnym czasowniku MQSUB, które utworzyło subskrypcję.
2. Jeśli temat subskrybowany przez oryginalny komendę MQSUB, który utworzył subskrypcję, zawiera znak wieloznaczny, może zostać wysłana więcej niż jedna zachowana publikacja. Liczba publikacji wysłanych w wyniku tego wywołania jest rejestrowana w polu NumPubs w strukturze Opts SubRq.
3. Jeśli komenda zakończy działanie z kodem przyczyny MQRC_NO_RETAINED_MSG, wówczas nie zachowano już publikacji dla podanego tematu. #
4. Jeśli komenda zakończy działanie z kodem przyczyny MQRC_RETAINED_MSG_Q_ERROR lub MQRC_RETAINED_NOT_DOSTARCZONYCH, wówczas istnieją publikacje zachowane dla określonego tematu, ale wystąpił błąd, który oznaczał, że nie można ich dostarczyć.
5. Aplikacja musi mieć bieżącą subskrypcję tematu, zanim będzie mogła wykonać to wywołanie. Jeśli subskrypcja została dokonana w poprzedniej instancji aplikacji, a poprawny uchwyt do subskrypcji nie jest dostępny, aplikacja musi najpierw wywołać MQSUB z opcją MQSO_RESUME, aby uzyskać uchwyt do użycia w tym wywołaniu.
6. Publikacje są wysyłane do miejsca docelowego, które jest zarejestrowane w celu użycia z bieżącą subskrypcją tej aplikacji. Jeśli publikacje muszą zostać wysłane w innym miejscu, należy najpierw zmodyfikować subskrypcję przy użyciu wywołania MQSUB z opcją MQSO_ALTER.

Wywołanie C

MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;      /* Connection handle */
MQHOBJ  Hsub;        /* Subscription handle */
MQLONG  Action;      /* Action requested by MQSUBRQ */
MQSRO   SubRqOpts;   /* Options that control the action of MQSUBRQ */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

Wywołanie języka COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Action requested by MQSUBRQ
01 ACTION PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
01 SUBRQOPTS.
COPY CMQSROV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn fixed bin(31); /* Connection handle */
dcl Hsub fixed bin(31); /* Subscription handle */
dcl Action fixed bin(31); /* Action requested by MQSUBRQ */
dcl SubRqOpts like MQSRO; /* Options that control the action of MQSUBRQ */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31); /* Reason code qualifying CompCode */
```

Wywołanie High Level Assembler

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN DS F Connection handle
HSUB DS F Subscription handle
ACTION DS F Action requested by MQSUBRQ
SUBRQOPTS CMQSROA , Options that control the action of MQSUBRQ
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

Atrybuty obiektów

Ta kolekcja tematów zawiera tylko te obiekty produktu IBM MQ , które mogą być przedmiotem wywołania funkcji MQINQ, a także podaje szczegółowe informacje na temat atrybutów, do których można się dowiedzieć, oraz do wybranych selektorów.

Atrybuty dla menedżera kolejek

Niektóre atrybuty menedżera kolejek są ustalane dla konkretnych implementacji; inne można zmienić za pomocą komendy MQSC ALTER QMGR.

Atrybuty te mogą być również wyświetlane za pomocą komendy DISPLAY QMGR. Większość atrybutów menedżera kolejek można uzyskać, otwierając specjalny obiekt MQOT_Q_MGR, a następnie za pomocą wywołania MQINQ z zwróconego uchwytu.

Poniższa tabela zawiera podsumowanie atrybutów, które są specyficzne dla menedżera kolejek. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy atrybutów wyświetlane w tej sekcji to nazwy opisowe używane w wywołaniu MQINQ. Nazwy te są takie same, jak w przypadku komend PCF. Jeśli komendy MQSC są używane do definiowania, modyfikowania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja [Komendy skryptowe \(MQSC\)](#).

Tabela 111. Atrybuty dla menedżera kolejek	
Atrybut	Opis
NadpiszAccountingConn	Nadpisz ustawienia rozliczeniowe.
AccountingInterval	Jak często zapisywać pośrednie rekordy rozliczeniowe.
NadpiszActivityConn	Nadpisz ustawienia działania.
ActivityTrace	Steruje gromadzeniem danych śledzenia działania aplikacji MQI produktu IBM MQ.
AdoptNewMCACheck	Elementy sprawdzane w celu określenia, czy ma zostać adopta nowa MCA.
AdoptNewMCAType	Określa, czy automatycznie restartować osierocone instancje agenta MCA określonego typu kanału.
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
AuthorityEvent	Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane).
BridgeEvent	Atrybut elementu sterującego dla zdarzeń mostu.
ChannelAutoDef	Określa, czy dozwolona jest automatyczna definicja kanału.
ChannelAutoDefEvent	Określa, czy generowane są zdarzenia automatycznego definiowania kanału.
ChannelAutoDefExit	Nazwa wyjścia użytkownika dla definicji kanału automatycznego
ChannelEvent	Atrybut elementu sterującego dla zdarzeń kanału.
Element sterujący ChannelInitiator	Atrybut elementu sterującego dla inicjatora kanału
ChannelMonitoring	Dane monitorowania online dla kanałów
ChannelStatistics	Steruje gromadzeniem danych statystycznych dla kanałów.
ChinitAdapters	Liczba podzadań adaptera do przetwarzania wywołań IBM MQ.
ChinitDispatchers	Liczba programów rozsyłających, które mają zostać użyte dla inicjatora kanału.
	Zarezerwowane do użycia przez produkt IBM.
ChinitTraceAutoStart	Określa, czy śledzenie inicjatora kanału powinno być uruchamiane automatycznie.
ChinitTraceTableSize	Wielkość przestrzeni danych śledzenia inicjatora kanału.
ClusterSenderMonitoringDefault	Domyślne dane monitorowania w trybie z połączeniem dla kanałów nadajnika klastrów
ClusterSenderStatystyki	Steruje gromadzeniem informacji o monitorowaniu statystyk dla kanałów nadajnika klastrów.
DaneClusterWorkload	Dane użytkownika dla wyjścia obciążenia klastra
ClusterWorkloadWyjście	Nazwa wyjścia użytkownika dla zarządzania obciążeniem klastra
ClusterWorkloadDługość	Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra
CLWLMRUChannels	Liczba ostatnio używanych kanałów dla równoważenia obciążenia klastra
CLWLUseQ	Obciążenie klastra korzysta z kolejki zdalnej.
CodedCharSetId	Identyfikator kodowanego zestawu znaków
CommandEvent	Atrybut elementu sterującego dla zdarzeń komendy.
CommandInput, atrybut QName	Nazwa kolejki wejściowej komend
CommandLevel	Poziom komendy
CommandServer, atrybut sterujący	Atrybut sterujący dla serwera komend.
atrybut zdarzenia konfiguracji	Atrybut elementu sterującego dla zdarzeń konfiguracji.
DeadLetter-nazwa QName	Nazwa kolejki niedostarczonych komunikatów
DEFCLXQ	Domyślny typ kolejki transmisji klastra

Tabela 111. Atrybuty dla menedżera kolejek (kontynuacja)

Atrybut	Opis
Nazwa QNameDefXmit	Domyślna nazwa kolejki transmisji
DistLists	Obsługa listy dystrybucyjnej
DNSGROUP	Nazwa grupy dla obiektu nasłuchiwanie TCP w przypadku korzystania z obsługi usług dynamicznych nazw domen menedżera obciążenia.
DNSWLM	Określa, czy program nasłuchujący TCP rejestruje się w programie Workload Manager for Dynamic Domain Name Services.
ExpiryInterval	Odstęp czasu między kolejnymi skanowaniami komunikatów, które
IGQPutAuthority	Uprawnienie do umieszczania w kolejkach wewnątrz grupy
IGQUserId	Identyfikator użytkownika kolejkowania wewnątrz grupy
InhibitEvent	Określa, czy mają być generowane zdarzenia zablokowanej (Inhibit Get and Inhibit Put)
IPAddressVersion	Wersja adresu Internet Protocol
IntraGroupqueuing	Obsługa kolejkowania wewnątrz grupy
ListenerTimer	Odstęp czasu między próbami zrestartowania obiektu nasłuchiwanie po awarii komunikacji APPC lub TCP/IP.
LocalEvent	Określa, czy generowane są lokalne zdarzenia błędów.
LoggerEvent	Określa, czy generowane są zdarzenia programu rejestrującego
LUGroupName	Ogólna nazwa LU dla programu nasłuchującego LU 6.2 obsługującego transmisje przychodzące dla grupy współużytkowania kolejki.
LUNAME	Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 .
LU62ARMSuffix	Przyrostek SYS1.PARMLIB , element APPCPMxx, który nominuje LUADD dla tego inicjatora kanału.
LU62Channels	Maksymalna liczba bieżących kanałów lub podłączonych klientów, które używają jednostki logicznej 6.2.
MaxActiveChannels	Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie.
MaxChannels	Maksymalna liczba bieżących kanałów.
MaxHandles	Maksymalna liczba uchwytów
MaxMsgDługość	Maksymalna długość komunikatu w bajtach
MaxPriority , atrybut	Maksymalny priorytet
MaxPropertiesDługość	Maksymalna długość danych właściwości w bajtach
MaxUncommittedkomunikatów	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy
Rozliczanie MQIAccounting	Steruje gromadzeniem informacji rozliczeniowych dla danych MQI.
Statystyki MQIStatistics	Steruje gromadzeniem informacji o monitorowaniu statystyk dla menedżera kolejek.
MsgMarkBrowseInterval	Odstęp czasu, po upływie którego menedżer kolejek może usunąć znacznik z przejrzanych komunikatów.
OutboundPortMin.	Program <i>OutboundPortMin</i> definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.
OutboundPortMax.	Program <i>OutboundPortMax</i> definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.
PerformanceEvent	Określa, czy generowane są zdarzenia związane z wydajnością.
Platforma	Platforma, na której działa menedżer kolejek
PubSubNPInputMsg	Informacja o tym, czy usunąć (lub zachować) niedostarczone komunikaty wejściowe
Odpowiedź NPPPubSub	Kontroluje zachowanie niedostarczone
PubSubMaxMsgRetryCount	Liczba ponowień podczas przetwarzania (w punkcie synchronizacji) komunikatu komendy zakończonej niepowodzeniem
PubSubSyncPoint	Określa, czy w punkcie synchronizacji mają być przetwarzane tylko trwałe (lub wszystkie) komunikaty
PubSubTryb	Określa, czy jest uruchomiony umieszczony w kolejce interfejs publikowania/subskrybowania
QMgrDesc	Opis menedżera kolejek
QMgrIdentifier	Unikalny wewnętrznie wygenerowany identyfikator menedżera kolejek
QMgrName	Nazwa menedżera kolejek

Tabela 111. Atrybuty dla menedżera kolejek (kontynuacja)

Atrybut	Opis
Nazwa QSGName	Nazwa grupy współużytkowania kolejki
QueueAccounting	Steruje kolekcją informacji rozliczeniowych dla kolejek.
QueueMonitoring	Dane monitorowania w trybie z połączeniem dla kolejek
QueueStatistics	Steruje gromadzeniem danych statystycznych dla kolejek.
ReceiveTimeout	Czas oczekiwania przez kanał TCP/IP na dane przed powrotem do stanu nieaktywnego.
ReceiveTimeoutMin	Kwalifikator dla <i>ReceiveTimeout</i> .
ReceiveTimeoutTyp	Minimalny czas, przez jaki kanał TCP/IP czeka na dane przed powrotem do stanu nieaktywnego.
RemoteEvent	Określa, czy generowane są zdalne zdarzenia błędów
RepositoryName	Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium
RepositoryNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium
ScyCase	Przypadek profili zabezpieczeń
NazwaSharedQMgr	Nazwa menedżera kolejek współużytkowanej kolejki
“SPLCAP” na stronie 841	IBM MQ Zaawansowane zabezpieczenie zabezpieczeń komunikatów dla menedżera kolejek włączonego lub wyłączonego.
SSLCRLNamelist 1	Nazwa obiektu listy nazw zawierającego nazwy obiektów informacji uwierzytelniającej.
SSLCryptoHardware 1	Łańcuch konfiguracji sprzętu szyfrującego.
SSEvent	Atrybut elementu sterującego dla zdarzeń SSL.
SSLFIPSREQUIRED	Używaj tylko algorytmów certyfikowanych przez FIPS dla kryptografii.
SSLKeyRepository 1	Położenie repozytorium kluczy SSL.
SSLKeyResetLiczba	Licznik zerowania klucza SSL.
Zadania SSLTasks 1	Liczba podzadań serwera na potrzeby przetwarzania wywołań SSL.
StatisticsInterval	Jak często zapisywać dane monitorowania statystyk.
StartStopZdarzenie	Określa, czy zdarzenia uruchomienia i zatrzymania są generowane
SyncPoint	Dostępność punktu synchronizacji
Kanały TCP	Maksymalna liczba bieżących kanałów lub podłączonych klientów, które używają protokołu TCP/IP.
TCPKeepAlive	Określa, czy użyć protokołu TCP KEEPALIVE do sprawdzenia innego końca połączenia.
TCPNAME	Nazwa systemu TCP/IP, który jest używany.
TCPStackType	Sposób, w jaki inicjator kanału może używać adresów TCP/IP.
TraceRouteRejestrowanie, atrybut	Steruje rejestrowaniem informacji o trasie śledzenia.
TriggerInterval	Przedział czasu komunikatu wyzwalacza
Wersja	Wersja
XrCapability	Określa, czy komendy telemetryczne są obsługiwane.
Uwagi: 1. Ten atrybut nie może zostać zapytany przy użyciu wywołania MQINQ i nie jest opisany w tej sekcji. Szczegółowe informacje na temat tego atrybutu zawiera sekcja Zmiana menedżera kolejek .	

Informacje pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

[FIPS \(Federal Information Processing Standards\) dla produktów UNIX, Linux i Windows](#)

Nadpisanie AccountingConn(MQLONG)

Pozwala to aplikacjom przesłonić ustawienie wartości ACCTMQI i ACCTQDATA w atrybucie Qmgr.

Wartość ta jest jedną z następujących wartości:

MQMON_DISABLED

Aplikacje nie mogą przestonić ustawienia atrybutów ACCTMQI i ACCTQ Qmgr, używając pola Opcje w strukturze MQCNO w wywołaniu MQCONN. Jest to wartość domyślna.

MQMON_ENABLED

Aplikacje mogą przestonić atrybuty ACCTQ i ACCTMQI Qmgr, używając pola Opcje w strukturze MQCNO.

Zmiany tej wartości są skuteczne tylko w przypadku połączeń z menedżerem kolejek po wprowadzeniu zmiany w atrybucie.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Unix i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_CONN_OVERRIDE przy użyciu wywołania MQINQ.

AccountingInterval (MQLONG)

Określa, jak długo przed zapisami pośrednich zapisów księgowych (w sekundach).

Wartość jest liczbą całkowitą z zakresu od 0 do 604800, przy czym wartość domyślna to 1800 (30 minut). Podaj wartość 0, aby wyłączyć rekordy pośrednie.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Windows, UNIX i Linux.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_INTERVAL przy użyciu wywołania MQINQ.

Nadpisanie ActivityConn(MQLONG)

Pozwala to aplikacjom przestonić ustawienie wartości ACTVTRC w atrybucie menedżera kolejek.

Wartość ta jest jedną z następujących wartości:

MQMON_DISABLED

Aplikacje nie mogą przestonić ustawienia atrybutu menedżera kolejek ACTVTRC przy użyciu pola Opcje w strukturze MQCNO w wywołaniu MQCONN. Jest to wartość domyślna.

MQMON_ENABLED

Aplikacje mogą przestonić atrybut menedżera kolejek ACTVTRC przy użyciu pola Opcje w strukturze MQCNO.

Zmiany tej wartości są skuteczne tylko w przypadku połączeń z menedżerem kolejek po wprowadzeniu zmiany w atrybucie.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Unix i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVITY_CONN_OVERRIDE w połączeniu z wywołaniem MQINQ.

ActivityTrace (MQLONG)

Ta opcja steruje gromadzeniem danych śledzenia działania aplikacji MQI produktu IBM MQ.

Wartość ta jest jedną z następujących wartości:

MQMON_ON

Zbierz dane śledzenia działań aplikacji MQI produktu IBM MQ.

MQMON_OFF,

Nie należy gromadzić danych śledzenia działania aplikacji MQI produktu IBM MQ. Jest to wartość domyślna.

Jeśli atrybut menedżera kolejek ACTVCON0 zostanie ustawiony na wartość ENABLED, ta wartość może zostać przestonięta dla pojedynczych połączeń, używając pola Opcje w strukturze MQCNO.

Zmiany tej wartości są skuteczne tylko w przypadku połączeń z menedżerem kolejek po wprowadzeniu zmiany w atrybucie.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Unix i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVITY_TRACE przy użyciu wywołania MQINQ.

AdoptNewMCACheck (MQLONG)

Definiuje to elementy, które mają zostać sprawdzone w celu określenia, czy agent MCA ma zostać adoptowany po wykryciu nowego kanału danych przychodzących o tej samej nazwie co agent MCA, który już jest aktywny.

Wartość ta jest jedną z następujących wartości:

MQADOPT_CHECK_Q_MGR_NAME,

Sprawdź nazwę menedżera kolejek.

MQADOPT_CHECK_NET_ADDR

Sprawdź adres sieciowy.

MQADOPT_CHECK_ALL

Sprawdź nazwę menedżera kolejek i adres sieciowy. Jeśli to możliwe, należy wykonać tę kontrolę, aby chronić kanały przed zamkniętym, nieumyślnie lub złośliwie. Jest to wartość domyślna.

MQADOPT_CHECK_NONE

Nie sprawdzaj żadnych elementów.

Zmiany wprowadzone w tym atrybucie zostaną zastosowane po następnym podjętym przez kanał próbie przyjęcia kanału.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ADOPTNEWMCA_CHECK z wywołaniem MQINQ.

AdoptNewMCAType (MQLONG)

Określa, czy po wykryciu nowego żądania kanału przychodzącego zgodnego z atrybutem MCACheck AdoptNewzostanie zrestartowany automatycznie osierocona instancja agenta MCA określonego typu kanału.

Jest to jedna z następujących wartości:

MQADOPT_TYPE_NO

Adoptowanie osieroconych instancji kanału nie jest wymagane. Jest to wartość domyślna.

MQADOPT_TYPE_ALL

Adoptować wszystkie typy kanałów.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ADOPTNEWMCA_TYPE z wywołaniem MQINQ.

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

Określa, czy zdarzenia autoryzacji (nie są autoryzowane) są generowane. Jest to jedna z następujących wartości:

MQEVN_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVN_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_AUTHORITY_EVENT z wywołaniem MQINQ.

BridgeEvent (MQLONG)

Określa, czy generowane są zdarzenia mostu IMS .

Wartość ta jest jedną z następujących wartości:

MQEVN_ENABLED

Wygeneruj zdarzenia mostu IMS w następujący sposób:

MQRC_BRIDGE_STARTED,

MQRC_BRIDGE_STOPPED

MQEVN_DISABLED

Nie generuj zdarzeń mostu IMS . Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_BRIDGE_EVENT przy użyciu wywołania MQINQ.

ChannelAutoDef (MQLONG)

Ten atrybut steruje automatycznym definiowaniem kanałów typu MQCHT_RECEIVER i MQCHT_SVRCONN. Automatyczna definicja kanałów MQCHT_CLUSSDR jest zawsze włączona. Wartość ta jest jedną z następujących wartości:

MQCHAD_DISABLED

Automatyczne definiowanie kanału zostało wyłączone.

MQCHAD_ENABLED

Włączono automatyczne definiowanie kanału.

Ten atrybut jest obsługiwany tylko w systemie AIX, HP-UX, IBM i, Linux, Solaris i Okna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_AUTO_DEF za pomocą wywołania MQINQ.

ChannelAutoDefEvent (MQLONG)

Służy do określania, czy generowane są zdarzenia automatycznego definiowania kanału. Ma zastosowanie do kanałów typu MQCHT_RECEIVER, MQCHT_SVRCONN i MQCHT_CLUSSDR. Wartość ta jest jedną z następujących wartości:

MQEVN_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVN_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Ten atrybut jest obsługiwany tylko w systemie AIX, HP-UX, IBM i, Linux, Solaris i Okna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_AUTO_DEF_EVENT przy użyciu wywołania MQINQ.

ChannelAutoDefExit (MQCHARn)

Jest to nazwa wyjścia użytkownika dla definicji kanału automatycznego. Jeśli ta nazwa jest niepusta, a parametr *ChannelAutoDef* ma wartość MQCHAD_ENABLED, to wyjście jest wywoływane za każdym razem, gdy menedżer kolejek ma utworzyć definicję kanału. Odnosi się to do kanałów typu MQCHT_RECEIVER, MQCHT_SVRCONN i MQCHT_CLUSSDR. Następnie program obsługi wyjścia może wykonać jedną z następujących czynności:

- Utwórz definicję kanału bez zmiany.
- Zmodyfikuj atrybuty definicji kanału, która została utworzona.
- Tłumić tworzenie kanału w całości.

Uwaga: Zarówno długość, jak i wartość tego atrybutu są specyficzne dla środowiska. Aby uzyskać szczegółowe informacje na temat wartości tego atrybutu w różnych środowiskach, należy zapoznać się z wprowadzeniem do struktury MQCD w produkcie [“MQCD-definicja kanału”](#) na stronie 1569 .

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windows i z/OS. W systemie z/OS ma zastosowanie tylko do kanałów wysyłających klastry i kanały odbierające klastry.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CHANNEL_AUTO_DEF_EXIT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)

Określa, czy generowane są zdarzenia kanału.

Jest to jedna z następujących wartości:

MQEVR_EXCEPTION

Generuj tylko następujące zdarzenia kanału:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED z następującym ReasonQualifiers:
 - MQRQ_CHANNEL_STOPPED_ERROR
 - MQRQ_CHANNEL_STOPPED_RETRY
 - MQRQ_CHANNEL_STOPPED_DISABLED
- MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED

Wygeneruj wszystkie zdarzenia kanału. Oznacza to, że oprócz tych wygenerowanych przez wyjątek EXCEPTION, generowane są następujące zdarzenia kanału:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED z następującym ReasonQualifier:
 - MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED

Nie generuj zdarzeń kanału. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_EVENT z wywołaniem MQINQ.

Element sterujący ChannelInitiator(MQLONG)

Określa, czy inicjator kanału ma być uruchamiany podczas uruchamiania menedżera kolejek.

Jest to jedna z następujących wartości:

Instrukcja MQSVC_CONTROL_MANUAL

Inicjator kanału nie może być uruchamiany automatycznie.

MQSVC_CONTROL_Q_MGR

Inicjator kanału ma być uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_CONTROL przy użyciu wywołania MQINQ.

ChannelMonitoring (MQLONG)

Ten atrybut określa dane monitorowania w trybie z połączeniem dla kanałów.

Wartość ta jest jedną z następujących wartości:

MQMON_NONE

Wyłącz gromadzenie danych na potrzeby monitorowania kanału dla wszystkich kanałów bez względu na ustawienie atrybutu kanału MONCHL. Jest to wartość domyślna.

MQMON_OFF,

Wyłącz gromadzenie danych monitorowania dla kanałów, które określają QMGR w atrybucie kanału MONCHL.

MQMON_LOW

Włącz gromadzenie danych monitorowania przy niskim współczynniku gromadzenia danych dla kanałów, w których określono QMGR w atrybucie kanału MONCHL.

MQMON_MEDIUM

Należy włączyć gromadzenie danych monitorowania przy użyciu umiarkowanego współczynnika gromadzenia danych dla kanałów, w których określono QMGR w atrybucie kanału MONCHL.

MQMON_HIGH

Należy włączyć gromadzenie danych monitorowania przy użyciu wysokiego współczynnika gromadzenia danych dla kanałów określających QMGR w atrybucie kanału MONCHL.

W systemie z/OS ten parametr po prostu włącza gromadzenie danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji LOW, MEDIUM lub HIGH nie ma wpływu na wyniki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_CHANNEL przy użyciu wywołania MQINQ.

ChannelStatistics (MQLONG)

Ta opcja steruje gromadzeniem danych statystycznych dla kanałów.

Wartość ta jest jedną z następujących wartości:

MQMON_NONE

Wyłącz gromadzenie danych dla statystyk kanału dla wszystkich kanałów bez względu na ustawienie atrybutu kanału STATCHL. Jest to wartość domyślna.

MQMON_OFF,

Wyłącz gromadzenie danych statystycznych dla kanałów, które określają QMGR w atrybucie kanału STATCHL.

MQMON_LOW

Włącz gromadzenie danych statystycznych z niskim współczynnikiem gromadzenia danych dla kanałów, w których określono QMGR w atrybucie kanału STATCHL.

MQMON_MEDIUM

Włącz gromadzenie danych statystycznych o umiarkowanym współczynniku gromadzenia danych dla kanałów, w których określono QMGR w atrybucie kanału STATCHL.

MQMON_HIGH

Włącz gromadzenie danych statystycznych o wysokim współczynniku gromadzenia danych dla kanałów, w których określono QMGR w atrybucie kanału STATCHL.

W przypadku większości systemów zaleca się użycie nośnika MEDIUM. Jednak w przypadku kanału, który przetwarza dużą liczbę komunikatów na sekundę, można zmniejszyć poziom próbkowania, wybierając opcję LOW. Ponadto w przypadku kanału, który przetwarza tylko kilka komunikatów i dla których najbardziej aktualne informacje są istotne, można wybrać wartość HIGH.

W systemie z/OS ten parametr po prostu włącza gromadzenie danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji LOW, MEDIUM lub HIGH nie ma wpływu na wyniki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_CHANNEL przy użyciu wywołania MQINQ.

ChinitAdapters (MQLONG)

Jest to liczba podzadań adaptera, które mają być używane do przetwarzania wywołań produktu IBM MQ. Wartość musi być z zakresu od 0 do 9999, a wartość domyślna to 8.

Stosunek liczby adapterów do przekaźników (atrybut ChinitDispatchers) powinien wynosić około 8 do 5. Jeśli jednak masz tylko kilka kanałów, to nie musisz zmniejszać wartości tego parametru z wartości domyślnej. Można użyć następujących wartości: dla systemu testowego, 8 (wartość domyślna), dla systemu produkcyjnego, 20. W idealnym przypadku należy mieć 20 adapterów, co zapewnia większy paralelizm wywołań produktu IBM MQ. Jest to istotne w przypadku komunikatów trwałych. Mniejsza liczba adapterów może być lepsza w przypadku komunikatów nietrwałych.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_ADAPTERS przy użyciu wywołania MQINQ.

ChinitDispatchers (MQLONG)

Jest to liczba programów rozsyłających, które mają być używane przez inicjatora kanału. Wartość musi być z zakresu od 0 do 9999, a wartość domyślna to 5.

Jako wytyczne można zezwolić na jeden przekaźnik dla 50 bieżących kanałów. Jeśli jednak masz tylko kilka kanałów, to nie musisz zmniejszać wartości tego atrybutu z wartości domyślnej. Jeśli używany jest protokół TCP/IP, największą liczbą programów rozsyłających, które są używane dla kanałów TCP/IP, jest 100, nawet jeśli w tym miejscu zostanie podana większa wartość. Można użyć następujących ustawień: systemy testowe, 5 (domyślnie); systemy produkcyjne, 20 (potrzeba 20 przekaźników do obsługi do 1000 aktywnych kanałów).

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_DISPATCHERS z wywołaniem MQINQ.

ChinitTraceAutoStart (MQLONG)

Określa, czy śledzenie inicjatora kanału ma być uruchamiane automatycznie.

Wartość ta jest jedną z następujących wartości:

MQTRAXSTR_YES

Automatycznie uruchom śledzenie inicjatora kanału. Jest to wartość domyślna.

MQTRAXSTR_NO

Nie uruchamiaj śledzenia inicjatora kanału automatycznie.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_TRACE_AUTO_START przy użyciu wywołania MQINQ.

ChinitTraceTableSize (MQLONG)

Jest to wielkość obszaru danych śledzenia inicjatora kanału (w MB).

Wartość musi mieścić się w zakresie od 0 do 2048, przy czym wartość domyślna to 2.

Uwaga: Za każdym razem, gdy używane są duże obszary danych produktu z/OS , należy upewnić się, że w systemie jest wystarczająca ilość pamięci dyskowej do obsługi wszystkich pokrewnych działań stronicowania serwera z/OS . Może także być konieczne zwiększenie wielkości zestawów danych SYS1.DUMP.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_TRACE_TABLE_SIZE z wywołaniem MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Określa wartość, która ma być podstawiana dla atrybutu ChannelMonitoring automatycznie zdefiniowanych kanałów nadajnika klastrów.

Wartość ta jest jedną z następujących wartości:

MQMON_Q_MGR

Gromadzenie danych monitorowania w trybie z połączeniem jest dziedziczone z ustawienia atrybutu *ChannelMonitoring* menedżera kolejek. Jest to wartość domyślna.

MQMON_OFF,

Monitorowanie kanału jest wyłączone

MQMON_LOW

Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z niskim wskaźnikiem gromadzenia danych przy minimalnym wpływie na wydajność systemu. Zgromadzone dane prawdopodobnie nie są najbardziej aktualne.

MQMON_MEDIUM

Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z umiarkowaną szybkością gromadzenia danych z ograniczonym wpływem na wydajność systemu.

MQMON_HIGH

Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z dużą szybkością gromadzenia danych, co może mieć wpływ na wydajność systemu. Zgromadzone dane są najbardziej aktualne.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_AUTO_CLUSSDR za pomocą wywołania MQINQ.

Statystyki ClusterSender(MQLONG)

Ponieważ kanały nadawcze klastra mogą być automatycznie definiowane na podstawie definicji CLUSRCVR w repozytorium, nie można zmienić ustawienia atrybutu STATCHL dla tych automatycznie zdefiniowanych kanałów nadawczych klastra za pomocą instrukcji ALTER channel. W przypadku tych kanałów decyzja o tym, czy gromadzić dane monitorowania w trybie z połączeniem, jest oparta na ustawieniu tego atrybutu menedżera kolejek.

Wartość ta jest jedną z następujących wartości:

MQMON_Q_MGR

Gromadzenie danych statystycznych dla automatycznie definiowanych kanałów nadajnika klastrów jest oparte na wartości atrybutu STATCHL menedżera kolejek. Jest to wartość domyślna.

MQMON_OFF,

Wyłącz kolekcjonowanie danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów.

MQMON_LOW

Przełącza gromadzenie danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów o niskim współczynniku gromadzenia danych.

MQMON_MEDIUM

Przełącza gromadzenie danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów o umiarkowanym współczynniku gromadzenia danych.

MQMON_HIGH

Przełącza gromadzenie danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów o wysokim współczynniku gromadzenia danych.

Dla większości systemów polecamy MEDIUM. Jednak w przypadku automatycznie zdefiniowanego kanału nadawczego klastra, który przetwarza dużą liczbę komunikatów na sekundę, można zmniejszyć poziom próbkowania, wybierając opcję LOW. Ponadto w przypadku kanału, który przetwarza tylko kilka komunikatów i dla których najbardziej aktualne informacje są istotne, można wybrać wartość HIGH.

W systemie z/OS ten parametr po prostu włącza gromadzenie danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji LOW, MEDIUM lub HIGH nie ma wpływu na wyniki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_AUTO_CLUSSDR przy użyciu wywołania MQINQ.

Dane ClusterWorkload(MQCHAR32)

Jest to 32-bajtowy łańcuch znaków zdefiniowany przez użytkownika, który jest przekazywany do wyjścia obciążenia klastra, gdy jest on wywoływany. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windows i z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_WORKLOAD_DATA z wywołaniem MQINQ.

ClusterWorkloadExit (MQCHARn)

Jest to nazwa wyjścia użytkownika do zarządzania obciążeniem klastra. Jeśli ta nazwa nie jest pusta, to wyjście jest wywoływane za każdym razem, gdy komunikat jest umieszczany w kolejce klastra lub przenoszony z jednej kolejki nadawczej klastra do innej. Wyjście może następnie zaakceptować instancję kolejki wybraną przez menedżer kolejek jako miejsce docelowe dla komunikatu lub wybrać inną instancję kolejki.

Uwaga: Zarówno długość, jak i wartość tego atrybutu są specyficzne dla środowiska.

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windows i z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_WORKLOAD_EXIT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_EXIT_NAME_LENGTH.

ClusterWorkloadLength (MQLONG)

Jest to maksymalna długość danych komunikatu przekazywana do wyjścia obciążenia klastra. Rzeczywista długość danych przekazywanych do wyjścia to minimum:

- Długość komunikatu.
- Atrybut *MaxMsgLength* menedżera kolejek.
- Atrybut *ClusterWorkloadLength*.

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windows i z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLUSTER_WORKLOAD_LENGTH z wywołaniem MQINQ.

CLWLMRUKanały (MQLONG)

Określa maksymalną liczbę najczęściej używanych kanałów klastra, które mają być brane pod uwagę do użycia przez algorytm wyboru obciążenia klastra.

Jest to wartość z zakresu od 1 do 999999999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_MRU_CHANNELS przy użyciu wywołania MQINQ.

CLWLUseQ (MQLONG)

Określa, czy mają być używane kolejki zdalne dla obciążenia klastra.

Wartość ta jest jedną z następujących wartości:

MQCLWL_USEQ_ANY

Należy używać zarówno kolejek lokalnych, jak i zdalnych.

MQCLWL_USEQ_LOCAL

Nie należy używać kolejek zdalnych. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_USEQ z wywołaniem MQINQ.

CodedCharSetId (MQLONG)

Definiuje zestaw znaków używany przez menedżer kolejek dla wszystkich pól łańcucha znaków zdefiniowanych w interfejsie MQI, takich jak nazwy obiektów oraz data i godzina utworzenia kolejki. Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach obiektów. Nie stosuje się do danych aplikacji przenoszonych w komunikacji. Wartość zależy od środowiska:

- W systemie z/OSwartość ta jest ustawiana na podstawie parametrów systemowych podczas uruchamiania menedżera kolejek. Wartością domyślną jest 500.
- W systemie Windowswartością jest podstawowa CODEPAGE użytkownika tworzący menedżer kolejek.
- W systemie IBM iwartość ta jest ustawiana w środowisku po pierwszym utworzeniu menedżera kolejek.
- W systemie Systemy UNIXjest to wartość domyślna CODESET dla ustawień narodowych użytkownika tworzący menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CODED_CHAR_SET_ID z wywołaniem MQINQ.

CommandEvent (MQLONG)

Określa, czy zdarzenia komendy są generowane w następujący sposób:

MQEVR_DISABLED

Nie generuj zdarzeń komendy. Jest to opcja domyślna.

MQEVR_ENABLED

Generuj zdarzenia komendy.

MQEVR_NO_DISPLAY

Zdarzenia komend są generowane dla wszystkich pomyslnych komend innych niż MQINQ.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_COMMAND_EVENT z wywołaniem MQINQ.

Nazwa QName komendy CommandInput(MQCHAR48)

Jest to nazwa kolejki wejściowej komend zdefiniowanej w menedżerze kolejek lokalnych. Jest to kolejka, do której użytkownicy mogą wysyłać komendy, jeśli są do tego upoważnieni. Nazwa kolejki zależy od środowiska:

- W systemie z/OSnazwą kolejki jest SYSTEM.COMMAND.INPUT; komendy MQSC i PCF mogą być do niego wysyłane. Szczegółowe informacje na temat komend PCF zawiera sekcja [Komendy MQSC](#) , aby uzyskać szczegółowe informacje na temat komend MQSC i [Definicje formatów komend programowalnych](#) .
- We wszystkich innych środowiskach nazwą kolejki jest SYSTEM.ADMIN.COMMAND.QUEUE, a tylko komendy PCF mogą być do niego wysyłane. Jednak komenda MQSC może zostać wysłana do tej kolejki, jeśli komenda MQSC została ujęta w komendzie PCF typu MQCMD_ESCAPE. Więcej informacji na temat komendy Escape zawiera sekcja [Escape](#) .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_COMMAND_INPUT_Q_NAME w wywołaniu MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Wskazuje to poziom komend sterujących systemem obsługiwanych przez menedżer kolejek. Może to być jedna z następujących wartości:

MQCMDL_LEVEL_600

Poziom 600 komend sterujących systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIXWersja 6.0
- IBM WebSphere MQ dla HP-UXWersja 6.0
- IBM WebSphere MQ dla iSeriesWersja 6.0
- IBM WebSphere MQ dla LinuxWersja 6.0
- IBM WebSphere MQ dla SolarisWersja 6.0
- IBM WebSphere MQ dla WindowsWersja 6.0
- IBM WebSphere MQ dla z/OSWersja 6.0

MQCMDL_LEVEL_700

Poziom 700 komend sterujących systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIXWersja 7.0
- IBM WebSphere MQ dla HP-UXWersja 7.0
- IBM WebSphere MQ dla IBM iWersja 7.0
- IBM WebSphere MQ dla LinuxWersja 7.0
- IBM WebSphere MQ dla SolarisWersja 7.0
- IBM WebSphere MQ dla WindowsWersja 7.0
- IBM WebSphere MQ dla z/OSWersja 7.0

MQCMDL_LEVEL_701

Poziom 701 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIXWersja 7.0.1
- IBM WebSphere MQ dla HP-UXWersja 7.0.1
- IBM WebSphere MQ dla IBM iWersja 7.0.1
- IBM WebSphere MQ dla LinuxWersja 7.0.1
- IBM WebSphere MQ dla SolarisWersja 7.0.1
- IBM WebSphere MQ dla WindowsWersja 7.0.1
- IBM WebSphere MQ dla z/OSWersja 7.0.1

MQCMDL_LEVEL_710

Poziom 710 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIXWersja 7.1
- IBM WebSphere MQ dla HP-UXWersja 7.1
- IBM WebSphere MQ dla IBM iWersja 7.1
- IBM WebSphere MQ dla LinuxWersja 7.1
- IBM WebSphere MQ dla SolarisWersja 7.1
- IBM WebSphere MQ dla WindowsWersja 7.1
- IBM WebSphere MQ dla z/OSWersja 7.1

MQCMDL_LEVEL_750

Poziom 750 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIX Wersja 7.5
- IBM WebSphere MQ dla HP-UX Wersja 7.5
- IBM WebSphere MQ dla IBM i Wersja 7.5
- IBM WebSphere MQ dla Linux Wersja 7.5
- IBM WebSphere MQ dla Solaris Wersja 7.5
- IBM WebSphere MQ dla Windows Wersja 7.5

MQCMDL_LEVEL_800

Poziom 800 komend sterujących systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM WebSphere MQ dla AIX Wersja 8.0
- IBM WebSphere MQ dla HP-UX Wersja 8.0
- IBM WebSphere MQ dla IBM i Wersja 8.0
- IBM WebSphere MQ dla Linux Wersja 8.0
- IBM WebSphere MQ dla Solaris Wersja 8.0
- IBM WebSphere MQ dla Windows Wersja 8.0
- IBM WebSphere MQ dla z/OS Wersja 8.0

V 8.0.0.2 MQCMDL_LEVEL_801

Poziom 801 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX Wersja 8.0.0.2
- IBM MQ for HP-UX Wersja 8.0.0.2
- IBM MQ for IBM i Wersja 8.0.0.2
- IBM MQ for Linux Wersja 8.0.0.2
- IBM MQ for Solaris Wersja 8.0.0.2

V 8.0.0.3 MQCMDL_LEVEL_802

Poziom 802 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX Wersja 8.0.0.3
- IBM MQ for HP-UX Wersja 8.0.0.3
- IBM MQ for IBM i Wersja 8.0.0.3
- IBM MQ for Linux Wersja 8.0.0.3
- IBM MQ for Solaris Wersja 8.0.0.3
- IBM MQ for Windows Wersja 8.0.0.3

Zestaw komend sterujących systemem, które odpowiadają konkretnej wartości atrybutu *CommandLevel*, zależy od wartości atrybutu *Platform*. Oba te komendy muszą być używane do decydowania o tym, które komendy sterujące systemem są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_COMMAND_LEVEL przy użyciu wywołania MQINQ.

Element sterujący CommandServer(MQLONG)

Określa, czy serwer komend ma być uruchamiany podczas uruchamiania menedżera kolejek.

Możliwe wartości:

Instrukcja MQSVC_CONTROL_MANUAL

Serwer komend nie może być uruchamiany automatycznie.

MQSVC_CONTROL_Q_MGR

Serwer komend ma być uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

Ten atrybut nie jest obsługiwany w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CMD_SERVER_CONTROL przy użyciu wywołania MQINQ.

ConfigurationEvent (MQLONG)

Określa, czy generowane są zdarzenia konfiguracji.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CONFIGURATION_EVENT przy użyciu wywołania MQINQ.

Możliwe wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Nazwa QName DeadLetter(MQCHAR48)

Jest to nazwa kolejki zdefiniowanej w lokalnym menedżerze kolejek jako kolejka niedostarczonych komunikatów (niedostarczonych komunikatów). Komunikaty są wysyłane do tej kolejki, jeśli nie mogą być kierowane do ich poprawnego miejsca docelowego.

Na przykład komunikaty są umieszczane w tej kolejce, gdy:

- Komunikat dociera do menedżera kolejek, który jest przeznaczony dla kolejki, która nie została jeszcze zdefiniowana w tym menedżerze kolejek.
- Komunikat dociera do menedżera kolejek, ale kolejka, do której jest przeznaczony, nie może jej odebrać, ponieważ możliwe jest:
 - Kolejka jest pełna
 - Żądania umieszczania żądań są zablokowane
 - Węzeł wysyłający nie ma uprawnień do umieszczania komunikatów w kolejce.

Aplikacje mogą również umieszczać komunikaty w kolejce niedostarczonych komunikatów.

Komunikaty raportów są traktowane w taki sam sposób, jak zwykłe komunikaty. Jeśli komunikat raportu nie może zostać dostarczony do kolejki docelowej (zwykle jest to kolejka określona w polu *ReplyToQ* w deskrytorze komunikatu oryginalnego komunikatu), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Uwaga: Komunikaty, które przeszły czas utraty ważności (patrz sekcja [MQMD-Wygaśnięcie pola](#)). **nie** są przysyłane do tej kolejki po ich odrzuceniu. Jednak komunikat raportu o utracie ważności (MQRO_EXPIRATION) jest nadal generowany i wysyłany do kolejki produktu *ReplyToQ* , o ile jest to wymagane przez aplikację wysyłającą.

Komunikaty nie są umieszczane w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów), gdy aplikacja, która wydała żądanie umieszczenia, została powiadomiona synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 (na przykład komunikat umieszczany w kolejce lokalnej, dla którego żądania umieszczenia są zablokowane).

Komunikaty w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) czasami zawierają dane komunikatu aplikacji z przedrostkiem struktury MQDLH. Ta struktura zawiera dodatkowe informacje, które wskazują, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Więcej informacji na temat tej struktury można znaleźć w sekcji [“MQDLH-nagłówek Dead-letter”](#) na stronie 349.

Ta kolejka musi być kolejką lokalną, z atrybutem *Usage* o wartości MQUS_NORMAL.

Jeśli menedżer kolejek nie obsługuje kolejki niedostarczonych komunikatów (niedostarczonych komunikatów) lub nie zdefiniowano jednego z nich, to nazwa ta jest pusta. Wszystkie menedżery kolejek produktu IBM MQ obsługują kolejkę niedostarczonych komunikatów (undelivered-message), ale domyślnie nie jest ona zdefiniowana.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie została zdefiniowana, pełna lub nie do użycia z jakiegoś innego powodu, komunikat, który zostałby przesłany przez agenta kanału komunikatów, zostanie zachowany w kolejce transmisji.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DEAD_LETTER_Q_NAME w wywołaniu MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

DefClusterXmitQueueTyp (MQLONG)

Atrybut DefClusterXmitQueueTyp określa, która kolejka transmisji jest wybierana domyślnie przez kanały wysyłające klastry w celu pobierania komunikatów z, do kanałów odbierających klastry.

Wartości atrybutu **DefClusterXmitQueueType** to MQCLXQ_SCTQ lub MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Wszystkie kanały nadawcze klastra wysyłają komunikaty z kolejki SYSTEM.CLUSTER.TRANSMIT.QUEUE. Identyfikator correlID komunikatów umieszczonych w kolejce transmisji wskazuje, do którego kanału nadawczego klastra ma zostać przekazany komunikat.

Atrybut SCTQ jest ustawiany po zdefiniowaniu menedżera kolejek. To zachowanie jest niejawne w wersjach produktu IBM WebSphere MQ starszych niż 7.5. W poprzednich wersjach atrybut menedżera kolejek DefClusterXmitQueueType był nieobecny.

MQCLXQ_CHANNEL

Każdy kanał nadawczy klastra wysyła komunikaty z innej kolejki transmisji. Każda kolejka transmisji jest tworzona jako stała kolejka dynamiczna z kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Jeśli atrybut menedżera kolejek, DefClusterXmitQueue, typ, jest ustawiony na wartość CHANNEL, Konfiguracja domyślna została zmieniona w taki sposób, że kanały nadawcze klastra zostały powiązane z poszczególnymi kolejkami transmisji klastra. Kolejki transmisji to trwałe kolejki dynamiczne utworzone na podstawie kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Każda kolejka transmisji jest powiązana z jednym kanałem nadawczym klastra. Ponieważ jeden kanał nadawczy klastra obsługuje kolejkę transmisji klastra, kolejka transmisji zawiera komunikaty dla tylko jednego menedżera kolejek w jednym klastrze. Istnieje możliwość skonfigurowania klastrów w taki sposób, aby każdy menedżer kolejek w klastrze zawierał tylko jedną kolejkę klastra. W takim przypadku ruch komunikatów z menedżera kolejek do każdej kolejki klastra jest przekazywany niezależnie z komunikatów do kolejki.

Aby wysłać zapytanie o wartość, należy wywołać komendę MQINQ lub wysłać komendę Menedżer kolejek zapytania (MQCMD_INQUIRE_Q_MGR) PCF, ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE. Aby zmienić tę wartość, należy wysłać komendę PCF menedżera kolejek zmian (Change Queue Manager-MQCMD_CHANGE_Q_MGR), ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE.

Odsyłacze pokrewne

“MQINQ-zapytanie o atrybuty obiektu” na stronie 714

Wywołanie funkcji MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Informacje pokrewne

[Zmiana menedżera kolejek](#)

Zapytaj menedżera kolejek

Nazwa QName DefXmit(MQCHAR48)

Jest to nazwa kolejki transmisji używanej do przesyłania komunikatów do zdalnych menedżerów kolejek, jeśli nie ma innego wskazania, do której kolejki transmisji należy użyć.

Jeśli nie ma domyślnej kolejki transmisji, nazwa jest całkowicie pusta. Początkowa wartość tego atrybutu jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DEF_XMIT_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Wskazuje to, czy lokalny menedżer kolejek obsługuje listy dystrybucyjne w wywołaniach MQPUT i MQPUT1. Jest to jedna z następujących wartości:

MQDL_SUPPORTED

Obsługiwane są listy dystrybucyjne.

MQDL_NOT_SUPPORTED

Listy dystrybucyjne nie są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DIST_LISTS z wywołaniem MQINQ.

Grupa DNSGroup (MQCHAR18)

Ten parametr nie jest już używany. Zapoznaj się z [WLM/DNS, który nie jest już obsługiwany](#).

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DNS_GROUP z wywołaniem MQINQ. Długość tego atrybutu jest podana przez parametr MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (MQLONG)

Ten parametr nie jest już używany. Zapoznaj się z [WLM/DNS, który nie jest już obsługiwany](#).

Wartość ta jest jedną z następujących wartości:

MQDNSWLM_YES

Ta wartość może być widoczna w menedżerze kolejek, który został zmigrowany z wcześniejszej wersji. Wartość jest ignorowana.

MQDNSWLM_NO

Jest to jedyna wartość obsługiwana przez menedżer kolejek.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DNS_WLM z wywołaniem MQINQ.

ExpiryInterval (MQLONG)

Wskazuje to częstotliwość, z jaką menedżer kolejek skanuje kolejki w poszukiwaniu komunikatów, które utraciły ważność. Jest to przedział czasu (w sekundach) z zakresu od 1 do 99 999 999 lub następująca wartość specjalna:

MQEXPI_OFF

Menedżer kolejek nie skanuje kolejek w poszukiwaniu komunikatów, które utraciły ważność.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_EXPIRY_INTERVAL z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

IGQPutAuthority (MQLONG)

Ten atrybut ma zastosowanie tylko wtedy, gdy lokalny menedżer kolejek jest elementem grupy współużytkowania kolejki. Wskazuje on typ sprawdzania uprawnień, który jest wykonywany, gdy lokalny wewnątrzgrupowy agent kolejkowania (agent IGQ) usuwa komunikat ze współużytkowanej kolejki transmisji i umieszcza komunikat w kolejce lokalnej. Wartość ta jest jedną z następujących wartości:

MQIGQPA_DEFAULT

Identyfikator użytkownika sprawdzony pod kątem autoryzacji jest wartością pola *UserIdentifier* w *oddzielnej* strukturze MQMD, która jest powiązana z komunikatem w przypadku, gdy komunikat znajduje się w współużytkowanej kolejce transmisji. Jest to identyfikator użytkownika programu, który umieścił komunikat w współużytkowanej kolejce transmisji i jest zwykle taki sam, jak identyfikator użytkownika, pod którym uruchomiony jest zdalny menedżer kolejek.

Jeśli profil RESLEVEL wskazuje, że ma być sprawdzany więcej niż jeden identyfikator użytkownika, sprawdzany jest również identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*).

MQIGQPA_CONTEXT

Identyfikator użytkownika sprawdzony pod kątem autoryzacji jest wartością pola *UserIdentifier* w *oddzielnej* strukturze MQMD, która jest powiązana z komunikatem w przypadku, gdy komunikat znajduje się w współużytkowanej kolejce transmisji. Jest to identyfikator użytkownika programu, który umieścił komunikat w współużytkowanej kolejce transmisji i jest zwykle taki sam, jak identyfikator użytkownika, pod którym uruchomiony jest zdalny menedżer kolejek.

Jeśli profil RESLEVEL wskazuje, że ma być sprawdzany więcej niż jeden identyfikator użytkownika, sprawdzany jest również identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*) i wartość pola *UserIdentifier* w *wbudowanym* strukturze MQMD. Ten ostatni identyfikator użytkownika to zwykle identyfikator użytkownika aplikacji, z której pochodzi komunikat.

MQIGQPA_ONLY_IGQ

Identyfikatorem użytkownika sprawdzonym pod kątem autoryzacji jest identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*).

Jeśli profil RESLEVEL wskazuje, że ma być sprawdzany więcej niż jeden identyfikator użytkownika, ten identyfikator użytkownika jest używany do wszystkich sprawdzeń.

MQIGQPA_ALTERNATE_OR_IGQ

Identyfikatorem użytkownika sprawdzonym pod kątem autoryzacji jest identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*).

Jeśli profil RESLEVEL wskazuje, że ma być sprawdzany więcej niż jeden identyfikator użytkownika, sprawdzana jest również wartość pola *UserIdentifier* w *wbudowanym* strukturze MQMD. Ten identyfikator użytkownika jest zwykle identyfikatorem użytkownika aplikacji, z którego pochodzi komunikat.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_IGQ_PUT_AUTHORITY z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

IGQUserId (MQLONG)

Ten atrybut ma zastosowanie tylko wtedy, gdy lokalny menedżer kolejek jest elementem grupy współużytkowania kolejek. Określa on identyfikator użytkownika, który jest powiązany z lokalnym agentem kolejkowania wewnątrz grupy (agent IGQ). Identyfikator ten jest jednym z identyfikatorów użytkowników, które mogą być sprawdzane pod kątem autoryzacji, gdy agent IGQ umieszcza komunikaty w kolejkach lokalnych. Rzeczywiste identyfikatory użytkowników są zależne od ustawienia atrybutu *IGQPutAuthority* oraz od opcji zabezpieczeń zewnętrznych.

Jeśli pole *IGQUserId* jest puste, z agentem IGQ nie jest powiązany żaden identyfikator użytkownika, a odpowiednia kontrola autoryzacji nie jest wykonywana (choć inne identyfikatory użytkowników mogą nadal być sprawdzane pod kątem autoryzacji).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_IGQ_USER_ID przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_USER_ID_LENGTH.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

InhibitEvent (MQLONG)

Ta opcja określa, czy są generowane zdarzenia zablokowanej (Inhibit Get and Inhibit Put). Wartość ta jest jedną z następujących wartości:

MQEVN_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVN_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_EVENT przy użyciu wywołania MQINQ.

W systemie z/OSnie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

IntraGroupqueuing (MQLONG)

Ten atrybut ma zastosowanie tylko wtedy, gdy lokalny menedżer kolejek jest elementem grupy współużytkowania kolejki. Wskazuje, czy kolejkovanie wewnątrz grupy jest włączone dla grupy współużytkowania kolejki. Wartość ta jest jedną z następujących wartości:

MQIGQ_WYŁĄCZONE

Wszystkie komunikaty przeznaczone dla innych menedżerów kolejek w grupie współużytkowania kolejek są przesyłane za pomocą konwencjonalnych kanałów.

MQIGQ_ENABLED

Komunikaty przeznaczone dla innych menedżerów kolejek w grupie współużytkowania kolejek są przesyłane przy użyciu współużytkowanej kolejki transmisji, jeśli spełniony jest następujący warunek:

- Długość danych komunikatu plus nagłówek transmisji nie przekracza 63 kB (64 512 bajtów).

Zaleca się, aby dla nagłówka transmisji przydzielono nieco więcej miejsca niż wielkość parametru MQXQH. W tym celu udostępniana jest stała wartość MQ_MSG_HEADER_LENGTH.

Jeśli warunek ten nie jest spełniony, komunikat jest przesyłany za pomocą konwencjonalnych kanałów.

Uwaga: Jeśli włączono kolejkovanie wewnątrz grupy, kolejność komunikatów przesyłanych przy użyciu współużytkowanej kolejki transmisji nie jest zachowywana w odniesieniu do tych przesyłanych przy użyciu kanałów konwencjonalnych.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INTRA_GROUP_queuing przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

IPAddressVersion (MQLONG)

Określa, która wersja adresu IP, IPv4 lub IPv6, jest używana.

Ten atrybut jest odpowiedni tylko dla systemów, które działają zarówno w systemach IPv4 , jak i w produkcie IPv6 , i mają wpływ tylko na kanały zdefiniowane jako posiadające *TransportType* z MQXPY_TCP, gdy spełniony jest jeden z następujących warunków:

- *ConnectionName* kanału jest nazwą hosta, która jest tłumaczona zarówno na adres IPv4 , jak i IPv6 , a jego parametr *LocalAddress* nie jest określony.
- *ConnectionName* i *LocalAddress* kanału są nazwami hostów tłumacznymi zarówno na adresy IPv4 , jak i IPv6 .

Możliwe wartości:

MQIPADDR_IPv4

IPv4 jest używany.

MQIPADDR_IPv6

IPv6 jest używany.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_IP_ADDRESS_VERSION z wywołaniem MQINQ.

ListenerTimer (MQLONG)

Jest to odstęp czasu (w sekundach) między kolejnymi próbami zrestartowania obiektu nasłuchiwanie przez program IBM MQ, jeśli wystąpiła awaria APPC lub TCP/IP. Wartość musi należeć do zakresu od 5 do 9999, przy czym wartość domyślna to 60.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LISTENER_TIMER przy użyciu wywołania MQINQ.

LocalEvent (MQLONG)

Służy do określania, czy generowane są lokalne zdarzenia błędów. Wartość ta jest jedną z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LOCAL_EVENT z wywołaniem MQINQ.

W systemie z/OSnie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

LoggerEvent (MQLONG)

Służy do określania, czy generowane są zdarzenia dziennika odtwarzania. Wartość ta jest jedną z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LOGGER_EVENT przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany tylko w systemie AIX, HP-UX, IBM i, Linux, Solaris i Okna.

LUGroupName (MQCHAR8)

Jest to ogólna nazwa LU programu nasłuchującego LU 6.2, który obsługuje transmisje przychodzące dla grupy współużytkowania kolejki. Jeśli ta nazwa nie zostanie pusta, nie będzie można używać tego obiektu nasłuchiwanie.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU_GROUP_NAME w wywołaniu MQINQ. Długość tego atrybutu jest podana przez wartość MQ_LU_NAME_LENGTH.

Nazwa LUName (MQCHAR8)

Jest to nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2. Ustaw tę wartość na tę samą jednostkę logiczną, która jest używana przez proces nasłuchujący na potrzeby transmisji danych przychodzących. Jeśli ta nazwa pozostanie pusta, używana jest domyślna jednostka

logiczna APPC/MVS. Jest to zmienna, a więc zawsze należy ustawić wartość LUName, jeśli używana jest wartość LU6.2.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Jest to przyrostek SYS1.PARMLIB składowa APPCPMxx, która mianuje LUADD dla tego inicjatora kanału. Komenda z/OS SET APPC=xx jest wydawana, gdy menedżer ARM restartuje inicjator kanału. Jeśli ta nazwa pozostanie pusta, nie zostanie wydana instrukcja SET APPC=xx.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU62_ARM_SUFFIX przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez parametr MQ_ARM_SUFFIX_LENGTH.

LU62Channels (MQLONG)

Jest to maksymalna liczba kanałów, które mogą być bieżące lub klienci, które mogą być podłączone, które korzystają z protokołu transmisji LU 6.2 .

Wartość musi mieścić się w zakresie od 0 do 9999, przy czym wartość domyślna to 200. Jeśli ta wartość zostanie ustawiona na zero, protokół transmisji LU 6.2 nie będzie używany.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LU62_CHANNELS przy użyciu wywołania MQINQ.

MaxActiveKanały (MQLONG)

Ten atrybut jest maksymalną liczbą kanałów, które mogą być *aktywne* w dowolnym momencie.

Wartością domyślną jest wartość podana dla atrybutu MaxChannels.

W przypadku systemu z/OS wartość musi być z zakresu od 1 do 9 999.

W przypadku wszystkich pozostałych platform wartość musi być z zakresu od 1 do 65 535.

Parametr **MaxActiveChannels** jest atrybutem menedżera kolejek tylko w systemie z/OS . Na pozostałych platformach **MaxActiveChannels** jest atrybutem w pliku qm.ini . Informacje na temat ustawiania atrybutu **MaxActiveChannels** na innych platformach znajdują się w sekcji [sekcje pliku konfiguracyjnego dla kolejkowania rozproszonego](#) .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVE_CHANNELS przy użyciu wywołania MQINQ .

Informacje pokrewne

[Stany kanału](#)

MaxChannels (MQLONG)

Ten atrybut jest maksymalną liczbą kanałów, które mogą być *bieżące* (w tym kanały połączenia z serwerem z połączonymi klientami).

W przypadku systemu z/OS wartość musi mieścić się w zakresie od 1 do 9 999, przy czym wartość domyślna to 200.

W przypadku wszystkich pozostałych platform wartość musi należeć do zakresu od 1 do 65 535, przy czym wartość domyślna to 100.

System, który jest zajęty obsługiwaniem połączeń z sieci, może potrzebować wyższej liczby niż ustawienie domyślne. Określ wartość, która jest poprawna dla danego środowiska, najlepiej, obserwując zachowanie systemu podczas testowania.

Parametr **MaxChannels** jest atrybutem menedżera kolejek tylko w systemie z/OS . Na pozostałych platformach **MaxChannels** jest atrybutem w pliku `qm.ini` . Informacje na temat ustawiania atrybutu **MaxChannels** na innych platformach znajdują się w sekcji sekcje pliku konfiguracyjnego dla kolejkowania rozproszonego .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_CHANNELS przy użyciu wywołania MQINQ .

Informacje pokrewne

Stany kanału

MaxHandles (MQLONG)

Jest to maksymalna liczba otwartych uchwytów, które mogą być używane jednocześnie przez dowolne zadanie. Każde pomyślne wywołanie MQOPEN dla jednej kolejki (lub dla obiektu, który nie jest kolejką) używa jednego uchwytu. Ten uchwyt stanie się dostępny do ponownego wykorzystania podczas zamykania obiektu. Jednak po otwarciu listy dystrybucyjnej każda kolejka na liście dystrybucyjnej przydziela osobny uchwyt, tak aby wywołanie MQOPEN używało tylu uchwytów, ile kolejek znajdujących się na liście dystrybucyjnej. Musi to być brane pod uwagę przy podejmowaniu decyzji o odpowiedniej wartości dla *MaxHandles* .

Wywołanie MQPUT1 wykonuje wywołanie MQOPEN w ramach przetwarzania. W wyniku tego wywołania MQPUT1 używa tylu uchwytów, co operacja MQOPEN, ale uchwytów są używane tylko przez czas trwania wywołania MQPUT1 .

W systemie z/OS zadanie *zadanie* oznacza zadanie CICS , zadanie MVS lub region zależny produktu IMS .

Wartość mieści się w zakresie od 1 do 999 999 999. Wartość domyślna jest określana przez środowisko:

- W systemie z/OS wartością domyślną jest 100.
- We wszystkich innych środowiskach wartością domyślną jest 256.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_UCHWYTÓW przy użyciu wywołania MQINQ.

MaxMsgDługość (MQLONG)

Jest to długość najdłuższego komunikatu *fizycznego* , który może obsłużyć menedżer kolejek. Ponieważ jednak atrybut menedżera kolejek produktu *MaxMsgLength* można ustawić niezależnie od atrybutu kolejki produktu *MaxMsgLength* , najdłuższy komunikat fizyczny, który może zostać umieszczony w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, aplikacja może umieścić komunikat *logiczny* , który jest dłuższy niż mniejszy z dwóch atrybutów produktu *MaxMsgLength* , ale tylko wtedy, gdy aplikacja określa flagę MQMF_SEGMENTATION_ALLOWED w deskryptorach MQMD. Jeśli ta opcja jest określona, górna granica długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zwykle ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym aplikacja jest uruchomiona, powodują zmniejszenie dolnego limitu.

Dolny limit dla atrybutu *MaxMsgLength* wynosi 32 kB (32 768 bajtów). Górny limit wynosi 100 MB (104 857 600 bajtów).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_MSG_LENGTH przy użyciu wywołania MQINQ.

MaxPriority (MQLONG)

Jest to maksymalny priorytet komunikatu obsługiwany przez menedżer kolejek. Priorytety są różne od zera (najniższy) do *MaxPriority* (najwyższy).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_PRIORITY z wywołaniem MQINQ.

MaxPropertiesLength (MQLONG)

Jest on używany do kontrolowania wielkości właściwości, które mogą przepływać z komunikatem. Obejmuje to zarówno nazwę właściwości w bajtach, jak i wielkość wartości właściwości również w bajtach.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_PROPERTIES_LENGTH przy użyciu wywołania MQINQ.

MaxUncommittedMsgs (MQLONG)

Jest to maksymalna liczba niezatwierdzonych komunikatów, które mogą istnieć w obrębie jednostki pracy. Liczba niezatwierdzonych komunikatów jest sumą następujących wartości od początku bieżącej jednostki pracy:

- Komunikaty umieszczane przez aplikację przy użyciu opcji MQPMO_SYNCPOINT
- Komunikaty pobrane przez aplikację przy użyciu opcji MQGMO_SYNCPOINT
- Komunikaty wyzwalacza i komunikaty raportu COA wygenerowane przez menedżer kolejek dla komunikatów umieszczonych za pomocą opcji MQPMO_SYNCPOINT
- Komunikaty raportu COD wygenerowane przez menedżera kolejek dla komunikatów pobranych z opcją MQGMO_SYNCPOINT

Następujące wartości *nie* są liczone jako komunikaty niezatwierdzone:

- Komunikaty umieszczane lub pobierane przez aplikację poza jednostką pracy
- Komunikaty wyzwalacza lub komunikaty raportu COA/COD wygenerowane przez menedżera kolejek w wyniku komunikatów umieszczanych lub pobieranych poza jednostką pracy
- Komunikaty raportu o utracie ważności wygenerowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat o utracie ważności jest określone w komunikacie o utracie ważności MQGMO_SYNCPOINT)
- Komunikaty o zdarzeniach generowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat zdarzenia spowodowało określony parametr MQPMO_SYNCPOINT lub MQGMO_SYNCPOINT).

Uwaga:

1. Komunikaty raportu o wyjątkach są generowane przez agenta kanału komunikatów (MCA) lub przez aplikację i są traktowane w taki sam sposób, jak zwykłe komunikaty umieszczane lub pobierane przez aplikację.
2. Jeśli komunikat lub segment jest umieszczany za pomocą opcji MQPMO_SYNCPOINT, liczba niezatwierdzonych komunikatów jest zwiększana o jeden niezależnie od tego, ile fizycznych komunikatów faktycznie wynika z operacji put. (Więcej niż jeden komunikat fizyczny może spowodować, że menedżer kolejek musi podzielić się komunikatem lub segmentem).
3. Jeśli lista dystrybucyjna jest umieszczana za pomocą opcji MQPMO_SYNCPOINT, liczba niezatwierdzonych komunikatów jest zwiększana o jeden *dla każdego wygenerowanego komunikatu fizycznego*. Może to być tak mały, jak jeden lub tak wielki, jak liczba miejsc docelowych na liście dystrybucyjnej.

Dolny limit dla tego atrybutu wynosi 1; górny limit to 999 999 999. Wartością domyślną jest 10000.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_UNCOMMITTED_MSGS przy użyciu wywołania MQINQ.

MQIAccounting (MQLONG)

Ta opcja steruje kolekcją informacji rozliczeniowych dla danych MQI.

Wartość ta jest jedną z następujących wartości:

MQMON_ON

Zbierz dane rozliczeniowe interfejsu API.

MQMON_OFF,

Nie zbieraj danych rozliczeniowych interfejsu API. Jest to wartość domyślna.

Jeśli atrybut ACCTCONO menedżera kolejek zostanie ustawiony na wartość ENABLED, wartość ta może zostać przesłonięta dla pojedynczych połączeń, korzystając z pola Opcje w strukturze MQCNO. Zmiany tej wartości są skuteczne tylko w przypadku połączeń z menedżerem kolejek, które występują po wprowadzeniu zmiany w atrybucie.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Systemy UNIX i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_MQI przy użyciu wywołania MQINQ.

MQIStatistics (MQLONG)

Ta opcja steruje gromadzeniem informacji o monitorowaniu statystyk dla menedżera kolejek.

Wartość ta jest jedną z następujących wartości:

MQMON_ON

Zbierz statystykę MQI.

MQMON_OFF,

Nie zbieraj statystyk MQI. Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemach IBM i, UNIX i Linux i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_MQI przy użyciu wywołania MQINQ.

MsgMarkBrowseInterval (MQLONG)

Przedział czasu w milisekundach, po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania.

Jest to odstęp czasu (w milisekundach), po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania.

Ten atrybut opisuje przedział czasu, dla którego komunikaty, które zostały oznaczone jako przejrane przez wywołanie MQGET, przy użyciu opcji get message MQGMO_MARK_BROWSE_CO_OP, mają pozostać oznaczone jako przeglądane.

Menedżer kolejek może automatycznie usunąć zaznaczenie przeglądanych komunikatów, które zostały oznaczone jako przejrane przez współpracujący zestaw uchwytów, gdy zostały one oznaczone przez więcej niż ten przybliżony przedział czasu.

Nie ma to wpływu na stan dowolnego komunikatu oznaczonego jako przeglądanie, który został uzyskany w wyniku wywołania metody MQGET, przy użyciu opcji get message MQGMO_MARK_BROWSE_HANDLE.

Maksymalna wartość to 999 999 999, a wartością domyślną jest 5000. Wartość specjalna -1 dla *MsgMarkBrowseInterval* reprezentuje nieograniczony przedział czasu.



Ostrzeżenie: Ta wartość nie powinna być niższa niż wartość domyślna 5000.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MSG_MARK_BROWSE_INTERVAL przy użyciu wywołania MQINQ.

OutboundPortMaks. (MQLONG)

Jest to najwyższy numer portu w zakresie, zdefiniowany przez parametr OutboundPortMin i OutboundPort, wartości maksymalnej liczby portów, które mają być używane do wiązania kanałów wychodzących.

Wartość jest liczbą całkowitą z zakresu od 0 do 65535 i musi być równa lub większa od wartości minimalnej OutboundPortMin. Wartością domyślną jest 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OUTBOUND_PORT_MAX przy użyciu wywołania MQINQ.

OutboundPortMin (MQLONG)

Jest to najniższy numer portu w zakresie, zdefiniowany przez parametr OutboundPortMin i OutboundPort, wartości maksymalnej liczby portów, które mają być używane do wiązania kanałów wychodzących.

Wartość ta jest liczbą całkowitą z zakresu od 0 do 65535 i musi być równa lub mniejsza od wartości maksymalnej OutboundPort. Wartością domyślną jest 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OUTBOUND_PORT_MIN z wywołaniem MQINQ.

PerformanceEvent (MQLONG)

Określa, czy generowane są zdarzenia związane z wydajnością. Jest to jedna z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PERFORMANCE_EVENT z wywołaniem MQINQ.

Platforma (MQLONG)

Wskazuje to system operacyjny, na którym działa menedżer kolejek:

MQPL_AIX

AIX (ta sama wartość jak MQPL_UNIX).

MQPL_APPLIANCE

IBM MQ Appliance

MVS MQPL_MVS

z/OS (ta sama wartość co MQPL_ZOS).

MQPL_NSK

HP Integrity NonStop Server.

MQPL_OS390

z/OS (ta sama wartość co MQPL_ZOS).

MQPL_OS400

IBM i.

MQPL_UNIX

Systemy UNIX.

MQPL_WINDOWS_NT

Windows stowarzyszonych.

Z_MQPL_ZOS

z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PLATFORM z wywołaniem MQINQ.

PubSubNPInputMsg (MQLONG)

Informacja o tym, czy usunąć lub zachować niedostarczone komunikaty wejściowe.

Wartość ta jest jedną z następujących wartości:

MQUNDELIVERED_DISCARD

Nietrwale komunikaty wejścia mogą być usuwane, jeśli nie można ich przetworzyć.

Jest to wartość domyślna.

MQUNDELIVERED_KEEP

Nietrwałe komunikaty wejścia nie będą usuwane, jeśli nie można ich przetworzyć. W tej sytuacji interfejs w kolejce publikowania/subskrypcji będzie kontynuował ponawianie procesu w odpowiednich odstępach czasu i nie będzie kontynuował przetwarzania kolejnych komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_NP_MSG z wywołaniem MQINQ.

PubSubNPResponse (MQLONG)

Kontroluje zachowanie niedostarczanych komunikatów odpowiedzi.

Wartość ta jest jedną z następujących wartości:

MQUNDELIVERED_NORMAL

Nietrwałe odpowiedzi, które nie mogą być umieszczone w kolejce odpowiedzi, są umieszczane w kolejce niedostarczanych komunikatów, jeśli nie można ich umieścić w kolejce DLQ, a następnie są usuwane.

MQUNDELIVERED_SAFE

Nietrwałe odpowiedzi, których nie można umieścić w kolejce odpowiedzi, są umieszczane w kolejce niedostarczonych komunikatów. Jeśli nie można ustawić odpowiedzi i nie można jej umieścić w kolejce DLQ, w kolejce interfejs publikowania/subskrypcji wycofa bieżącą operację, a następnie ponów próbę w odpowiednich odstępach czasu i nie będzie kontynuować przetwarzania kolejnych komunikatów.

MQUNDELIVERED_DISCARD

Odpowiedzi nietrwałe nie są umieszczane w kolejce odpowiedzi są odrzucane.

Jest to wartość domyślna dla nowych menedżerów kolejek.

MQUNDELIVERED_KEEP

Odpowiedzi nietrwałe nie są umieszczane w kolejce niewysłanych wiadomości ani odrzucane. Zamiast tego w kolejce interfejs publikowania/subskrypcji wycofa bieżącą operację, a następnie ponów próbę w odpowiednich odstępach czasu.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_NP_RESP przy użyciu wywołania MQINQ.

Wartość domyślna dla migrowanych menedżerów kolejek.

Jeśli menedżer kolejek został zmigrowany z programu IBM MQ V6.0, początkowa wartość tego atrybutu zależy od wartości *DiscardNonPersistentResponse* i *DLQNonPersistentResponse* przed migracją, tak jak przedstawiono to w poniższej tabeli.

		Odpowiedź DLQNonPersistent		
		Tak	Nie	Nie ustawiono
DiscardNonPersistentResponse	Tak	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD	MQUNDELIVERED_NORMAL
	Nie	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	Nie ustawiono	Jeśli SyncPointPersistent = No, MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL	Jeśli SyncPointPersistent = No, MQUNDELIVERED_KEEP else MQUNDELIVERED_DISCARD	Jeśli SyncPointPersistent = No, MQUNDELIVERED_SAFE else MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

Liczba ponowień podczas przetwarzania komunikatu komendy zakończonej niepowodzeniem w punkcie synchronizacji.

Wartość ta jest jedną z następujących wartości:

0 - 999 999 999

Wartość domyślna to 5.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_MAXMSG_RETRY_COUNT z wywołaniem MQINQ.

PubSubSyncPoint (MQLONG)

Określa, czy tylko komunikaty trwałe lub wszystkie komunikaty są przetwarzane w punkcie synchronizacji.

Wartość ta jest jedną z następujących wartości:

IFSYNCPOINT_IFPER

Powoduje to, że w kolejce interfejs publikowania/subskrypcji odbiera komunikaty nietrwałe poza punktem synchronizacji. Jeśli demon odbierze publikację spoza punktu synchronizacji, przekazuje ją do znanych subskrybentów znajdujących się poza punktem synchronizacji.

Jest to wartość domyślna.

MQSYNCPOINT_YES

Powoduje to, że w kolejce interfejs publikowania/subskrypcji odbierze wszystkie komunikaty w punkcie synchronizacji.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_SYNC_PT przy użyciu wywołania MQINQ.

Tryb PubSub(MQLONG)

Określa, czy działa mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrybowania, umożliwiając aplikacjom publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez interfejs w kolejce publikowania/subskrypcji.

Wartość ta jest jedną z następujących wartości:

MQPSM_COMPAT

Mechanizm publikowania/subskrybowania działa. Dlatego możliwe jest publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego. Umieszczony w kolejce interfejs publikowania/subskrybowania nie jest uruchomiony, dlatego żaden komunikat umieszczony w kolejkach monitorowanych przez wstawiony interfejs publikowania/subskrybowania nie jest działął. To ustawienie jest używane w celu zapewnienia zgodności z produktem WebSphere Message Broker V6 lub wcześniejszymi wersjami za pomocą tego menedżera kolejek, ponieważ musi on odczytywać te same kolejki, z których normalnie jest odczytywać umieszczony w kolejce interfejs publikowania/subskrypcji.

MQPSM_DISABLED

Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania nie działają. Nie jest więc możliwe publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego. Wszystkie komunikaty publikowania/subskrybowania, które są umieszczane w kolejkach monitorowanych przez interfejs w kolejce publikowania/subskrypcji, nie są wykonywane.

MQPSM_ENABLED

Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania działają. Dlatego możliwe jest publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez interfejs w kolejce publikowania/subskrypcji. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_MODE z wywołaniem MQINQ.

QMGrDesc (MQCHAR64)

To pole służy do opisywania komentarza dotyczącego menedżera kolejek. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole to może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

- W systemie z/OS wartością domyślną jest nazwa produktu i numer wersji.
- We wszystkich innych środowiskach wartością domyślną jest odstępy.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_DESC przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_MGR_DESC_LENGTH.

QMGrIdentifier (MQCHAR48)

Jest to unikalna nazwa, która jest generowana jako unikalna dla menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_IDENTIFIER przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_MGR_IDENTIFIER_LENGTH.

Ten atrybut jest obsługiwany w następujących środowiskach: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows oraz klienty IBM MQ połączone z tymi systemami.

QMGrName (MQCHAR48)

Jest to nazwa lokalnego menedżera kolejek, tj. nazwa menedżera kolejek, z którym połączona jest aplikacja.

Pierwsze 12 znaków nazwy jest używane do konstruowania unikalnego identyfikatora komunikatu (patrz MQMD- MsgId field). Menedżery kolejek, które mogą wzajemnie się komunikować, muszą mieć nazwy różniące się od pierwszych 12 znaków, tak aby identyfikatory komunikatów były unikalne w sieci menedżera kolejek.

W systemie z/OS nazwa jest taka sama, jak nazwa podsystemu, która jest ograniczona do 4 niepustych znaków.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_NAME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

QSGName (MQCHAR4)

Jest to nazwa grupy współużytkowania kolejki, do której należy lokalny menedżer kolejek. Jeśli lokalny menedżer kolejek nie należy do grupy współużytkowania kolejek, nazwa jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_QSG_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_QSG_NAME_LENGTH.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

QueueAccounting (MQLONG)

Ta opcja steruje kolekcją informacji rozliczeniowych dla kolejek.

Wartość ta jest jedną z następujących wartości:

MQMON_NONE

Nie zbieraj danych rozliczeniowych dla kolejek, niezależnie od ustawienia atrybutu rozliczania kolejki ACCTQ. Jest to wartość domyślna.

MQMON_OFF,

Nie należy gromadzić danych rozliczeniowych dla kolejek, które określają QMGR w atrybucie kolejki ACCTQ.

MQMON_ON

Zbierz dane rozliczeniowe dla kolejek, które określają QMGR w atrybucie kolejki ACCTQ.

Zmiany tej wartości są skuteczne tylko w przypadku połączeń z menedżerem kolejek, które występują po wprowadzeniu zmiany w atrybucie.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_Q z wywołaniem MQINQ.

QueueMonitoring (MQLONG)

Określa domyślne ustawienie monitorowania kolejek w trybie z połączeniem.

Jeśli atrybut kolejki *QueueMonitoring* jest ustawiony na wartość *MQMON_Q_MGR*, ten atrybut określa wartość, która jest przyjmowana przez kanał. Możliwe wartości:

MQMON_OFF,

Gromadzenie danych monitorowania otwartej bazy danych jest wyłączone. Jest to początkowa wartość domyślna menedżera kolejek.

MQMON_NONE

Gromadzenie danych monitorowania w trybie z połączeniem jest wyłączone dla kolejek niezależnie od ustawienia ich atrybutu *QueueMonitoring*.

MQMON_LOW

Gromadzenie danych monitorowania w trybie z połączeniem jest włączone, przy niskim współczynniku gromadzenia danych.

MQMON_MEDIUM

Gromadzenie danych monitorowania w trybie z połączeniem jest włączone, a średni współczynnik gromadzenia danych jest umiarkowany.

MQMON_HIGH

Gromadzenie danych monitorowania w trybie z połączeniem jest włączone, przy wysokim współczynniku gromadzenia danych.

Aby określić wartość tego atrybutu, należy użyć selektora *MQIA_MONITORING_Q* z wywołaniem *MQINQ*.

QueueStatistics (MQLONG)

Ta opcja steruje gromadzeniem danych statystycznych dla kolejek.

Jest to jedna z następujących wartości:

MQMON_NONE

Nie zbieraj statystyk kolejek dla kolejek, niezależnie od ustawienia atrybutu kolejki *QueueStatistics*. Jest to wartość domyślna.

MQMON_OFF,

Nie należy gromadzić danych statystycznych dla kolejek, które określają menedżera kolejek w atrybucie kolejki *QueueStatistics*.

MQMON_ON

Zbierz dane statystyczne dla kolejek, które określają menedżera kolejek w atrybucie kolejki *QueueStatistics*.

Aby określić wartość tego atrybutu, należy użyć selektora *MQIA_STATISTICS_Q* z wywołaniem *MQINQ*.

ReceiveTimeout (MQLONG)

Określa, jak długo kanał TCP/IP oczekuje na otrzymywanie danych, w tym pulsy, od swojego partnera przed powrotem do stanu nieaktywnego. Ma ona zastosowanie tylko do kanałów komunikatów, a nie do kanałów MQI.

Dokładne znaczenie parametru *ReceiveTimeout* jest zmieniane przez wartość określoną w polu *ReceiveTimeoutType*. Typ *ReceiveTimeout* może być ustawiony na jeden z następujących:

- *MQRCVTIME_EQUAL*-ta wartość jest liczbą w sekundach, przez którą kanał ma czekać. Podaj wartość z zakresu od 0 do 999999.
- *MQRCVTIME_ADD*-ta wartość jest liczbą w sekundach, która ma zostać dodana do wynegocjowanego obiektu HBINT, i określa czas oczekiwania kanału. Podaj wartość z zakresu od 1 do 999999.
- *MQRCVTIME_MULTIPLY*-ta wartość jest mnożnikiem, który ma zostać zastosowany do wynegocjowanego obiektu HBINT. Podaj wartość 0 lub wartość z zakresu od 2 do 99.

Wartością domyślną jest 0.

Ustaw wartość parametru ReceiveTimeoutna wartość MQRCVTIME_MULTIPLY lub MQRCVTIME_EQUAL, a parametr ReceiveTimeout na wartość 0, aby zatrzymać kanał przed upływem czasu oczekiwania na odebranie danych od partnera.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT przy użyciu wywołania MQINQ.

ReceiveTimeoutMin (MQLONG)

Jest to minimalny czas (w sekundach), przez jaki kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od swojego partnera, przed powrotem do stanu nieaktywnego.

Ma ona zastosowanie tylko do kanałów komunikatów, a nie do kanałów MQI. Wartość musi być z zakresu od 0 do 999999, z wartością domyślną jest 0.

Jeśli używany jest typ ReceiveTimeout, aby określić, że czas oczekiwania kanału TCP/IP ma być obliczony względem wynegocjowanej wartości parametru HBINT, a wartość wynikowa jest mniejsza niż wartość tego parametru, ta wartość zostanie użyta.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT_MIN z wywołaniem MQINQ.

Typ ReceiveTimeout(MQLONG)

Jest to kwalifikator stosowany do metody ReceiveTimeout w celu zdefiniowania, jak długo kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od partnera, przed powrotem do stanu nieaktywnego. Ma ona zastosowanie tylko do kanałów komunikatów, a nie do kanałów MQI.

Wartość ta jest jedną z następujących wartości:

MQRCVTIME_MULTIPLY

Parametr ReceiveTimeout jest mnożnikiem, który ma zastosowanie do wynegocjowanej wartości HBINT, aby określić, jak długo kanał czeka. Jest to wartość domyślna.

MQRCVTIME_ADD,

ReceiveTimeout to wartość (w sekundach), która ma zostać dodana do wynegocjowanej wartości HBINT w celu określenia, jak długo kanał oczekuje.

MQRCVTIME_EQUAL

ReceiveTimeout to wartość (w sekundach), przez którą kanał oczekuje.

Aby zatrzymać czas oczekiwania przez kanał oczekiwania na odebranie danych od partnera, należy ustawić wartość parametru ReceiveTimeoutna wartość MQRCVTIME_MULTIPLY lub MQRCVTIME_EQUAL, a wartość ReceiveTimeout na wartość 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT_TYPE z wywołaniem MQINQ.

RemoteEvent (MQLONG)

Określa, czy generowane są zdalne zdarzenia błędów. Jest to jedna z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_REMOTE_EVENT z wywołaniem MQINQ.

RepositoryName (MQCHAR48)

Jest to nazwa klastra, dla którego ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę dla więcej niż jednego klastra, *RepositoryNameList* określa nazwę obiektu listy nazw, która identyfikuje klastry, a *RepositoryName* jest pusta. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNameList* musi być pusta.

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windowsi z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REPOSITORY_NAME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

RepositoryNameList (MQCHAR48)

Jest to nazwa obiektu listy nazw, który zawiera nazwy klastrów, dla których ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę tylko dla jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć opcji *RepositoryName* do określenia nazwy klastra, w którym to przypadku pole *RepositoryNameList* jest puste. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNameList* musi być pusta.

Ten atrybut jest obsługiwany tylko w systemach AIX, HP-UX, IBM i, Linux, Solaris, Windowsi z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REPOSITORY_NAMELIST przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Określa, czy menedżer kolejek obsługuje nazwy profili zabezpieczeń w przypadku mieszanym, czy tylko wielkimi literami.

Wartość ta jest jedną z następujących wartości:

MQSCYC_GÓRNY

Nazwy profili zabezpieczeń muszą być pisane wielkimi literami.

MQSCYC_MIESZANY

Nazwy profili zabezpieczeń mogą być pisane wielkimi literami lub literami o różnej wielkości.

Zmiany wprowadzone w tym atrybucie są aktywne po uruchomieniu komendy Refresh Security z podaną wartością *SecurityType (MQSECTYPE_CLASSES)*.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SECURITY_CASE z wywołaniem MQINQ.

Nazwa SharedQMGr(MQLONG)

Określa, czy produkt *ObjectQmgrName* powinien być używany lub traktowany jako lokalny menedżer kolejek w wywołaniu MQOPEN, dla kolejki współużytkowanej, gdy *ObjectQmgrName* jest kolejką współużytkowaną innego menedżera kolejek w grupie współużytkowania kolejki.

Możliwe wartości:

MQSQQM_USE

ObjectQmgrName jest używana i otwarta jest odpowiednia kolejka transmisji.

MQSQQM_IGNORE

Jeśli kolejka docelowa jest współużytkowana, a serwer *ObjectQmgrName* jest menedżerem kolejek w tej samej grupie współużytkowania kolejki, to operacja otwarcia jest wykonywana lokalnie.

Ten atrybut jest poprawny tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SHARED_Q_Q_MGR_NAME w wywołaniu MQINQ.

SPLCAP

Wskazuje, czy funkcje zabezpieczeń produktu IBM MQ Advanced Message Security są dostępne dla menedżera kolejek.

MQCAP_SUPPORTED

Jest to wartość domyślna, jeśli komponent IBM MQ AMS jest zainstalowany dla instalacji, w której jest uruchomiony menedżer kolejek.

MQCAP_NOT_SUPPORTED

SSLEvent (MQLONG)

Określa, czy zdarzenia SSL są generowane.

Jest to jedna z następujących wartości:

MQEVR_ENABLED

Wygeneruj zdarzenia SSL w następujący sposób:

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED

Nie generuj zdarzeń SSL. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_EVENT z wywołaniem MQINQ.

SSLFIPSRequired (MQLONG)

Umożliwia to określenie, że tylko algorytmy certyfikowane przez FIPS mają być używane, jeśli kryptografia jest wykonywana w produkcie IBM MQ, a nie w sprzęcie szyfrującym. Jeśli sprzęt szyfrujący jest skonfigurowany, używane moduły kryptograficzne to te moduły udostępniane przez produkt sprzętowy. Moduły te mogą lub nie muszą być certyfikowane przez FIPS na określonym poziomie w zależności od używanego produktu sprzętowego.

Wartość jest jedną z następujących wartości:

MQSSL_FIPS_NO

Użyj dowolnej opcji CipherSpec obsługiwanych przez platformę w użyciu. Ta wartość jest wartością domyślną.

MQSSL_FIPS_YES

W przypadku wszystkich połączeń SSL z i do tego menedżera kolejek należy używać tylko algorytmów szyfrujących zgodnych ze standardem FIPS w obszarze CipherSpecs .

Ten parametr jest poprawny tylko na platformach UNIX, Linux, Windows i z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_FIPS_REQUIRED z wywołaniem MQINQ.

Informacje pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

FIPS (Federal Information Processing Standards) dla produktów UNIX, Linux i Windows

Liczba operacji SSLKeyReset(MQLONG)

Określa, kiedy agenci kanału komunikatów kanału SSL (MCAs) inicjujący komunikację resetują klucz tajny używany do szyfrowania w kanale.

Wartość reprezentuje całkowitą liczbę nieszyfrowanych bajtów, które są wysyłane i odbierane za pomocą kanału przed renegotiacją klucza tajnego. Liczba bajtów obejmuje informacje sterujące wysłane przez agenta MCA.

Wartość jest liczbą z zakresu od 0 do 999 999 999, z wartością domyślną równą 0. Jeśli zostanie określona wartość resetowania klucza tajnego SSL/TLS w zakresie od 1 bajtu do 32 kB, kanały SSL/TLS będą używać klucza tajnego resetowania klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernych resetów klucza, które nastąpiłyby w przypadku małych wartości resetowania klucza tajnego SSL/TLS.

Klucz tajny jest renegotjowany, gdy łączna liczba niezaszyfrowanych bajtów wysłanych i odebranych przez kanał inicjujący MCA przekracza określoną wartość lub jeśli pulsy kanału są włączone, zanim dane zostaną wysłane lub odebrane po wystąpieniu pulsu kanału, w zależności od tego, co nastąpi wcześniej.

Liczba bajtów wysłanych i odebranych dla renegotjacji obejmuje informacje sterujące wysłane i odebrane przez kanał MCA kanału i jest resetowane za każdym razem, gdy wystąpi renegotjacja.

Należy użyć wartości 0, aby wskazać, że klucze tajne nigdy nie są renegotjowane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_RESET_COUNT z wywołaniem MQINQ.

Zdarzenie StartStop(MQLONG)

Określa, czy zdarzenia uruchomienia i zatrzymania są generowane. Wartość ta jest jedną z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_START_STOP_EVENT przy użyciu wywołania MQINQ.

StatisticsInterval (MQLONG)

Określa, jak często (w sekundach) zapisywać dane monitorowania statystyk w kolejce monitorowania.

Wartość jest liczbą całkowitą z zakresu od 0 do 604800, przy czym wartość domyślna to 1800 (30 minut).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_INTERVAL przy użyciu wywołania MQINQ.

SyncPoint (MQLONG)

Wskazuje to, czy lokalny menedżer kolejek obsługuje jednostki pracy i syncwskazujących wywołania MQGET, MQPUT i MQPUT1.

MQSP_AVAILABLE

Jednostki pracy i metody synchronizacji dostępne.

MQSP_NOT_AVAILABLE

Jednostki pracy i syncwskazujący nie są dostępne.

- W systemie z/OS ta wartość nigdy nie jest zwracana.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SYNCPOINT z wywołaniem MQINQ.

Kanały TCP (MQLONG)

Jest to maksymalna liczba kanałów, które mogą być bieżące, lub klientów, które mogą być podłączone, które korzystają z protokołu transmisji TCP/IP.

Wartość musi mieścić się w zakresie od 0 do 9999, przy czym wartość domyślna to 200. Jeśli zostanie podana wartość 0, protokół TCP/IP nie będzie używany.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_CHANNELS przy użyciu wywołania MQINQ.

TCPKeepAlive (MQLONG)

Określa, czy ma być używany protokół TCP KEEPALIVE w celu sprawdzenia, czy drugi koniec połączenia jest nadal dostępny. Jeśli drugi koniec połączenia nie jest dostępny, kanał zostanie zamknięty.

Wartość ta jest jedną z następujących wartości:

MQTCPKEEP_YES

Użyj protokołu TCP KEEPALIVE zgodnie z podanym w zestawie danych konfiguracyjnych profilu TCP. Jeśli zostanie określony atrybut kanału KeepAliveInterval (KAINT), zostanie użyta wartość, do której zostanie ustawiona wartość.

MQTCPKEEP_NO

Nie należy używać protokołu TCP KEEPALIVE. Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_KEEP_ALIVE przy użyciu wywołania MQINQ.

TCPName (MQCHAR8)

Jest to nazwa jedynego lub preferowanego stosu TCP/IP, który będzie używany, w zależności od wartości parametru TCPStackType. Ten parametr ma zastosowanie tylko w wielu środowiskach stosu CINET. Wartością domyślną jest TCP/IP.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_TCP_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_TCP_NAME_LENGTH.

TCPStackType (MQLONG)

Określa, czy inicjator kanału może używać tylko stosu TCP/IP określonego w nazwie TCPName, czy też może być przyłączony do dowolnego wybranego stosu TCP/IP. Ten parametr ma zastosowanie tylko w wielu środowiskach stosu CINET.

Wartość ta jest jedną z następujących wartości:

MQTCPSTACK_SINGLE

Inicjator kanału może używać tylko przestrzeni adresowych TCP/IP nazwanych w TCPName. Jest to wartość domyślna.

MQTCPSTACK_MULTIPLE

Inicjator kanału może korzystać z dowolnej dostępnej przestrzeni adresowej TCP/IP. Wartością domyślną jest wartość podana w nazwie TCPName, jeśli dla kanału lub obiektu nasłuchiwania nie określono żadnej innej wartości.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_STACK_TYPE z wywołaniem MQINQ.

TraceRouteRejestrowanie (MQLONG)

Ta opcja steruje rejestrowaniem informacji o trasie śledzenia.

Wartość ta jest jedną z następujących wartości:

MQRECORDING_DISABLED

Brak możliwości dopisania do komunikatów śledzenia trasy.

MQRECORDING_Q

Umieszczanie komunikatów śledzenia trasy do stałej kolejki nazwanej.

MQRECORDING_MSG

Komunikaty śledzenia trasy są umieszczane w kolejce, która jest określana przy użyciu samego komunikatu. Jest to wartość domyślna

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRACE_ROUTE_RECORDING przy użyciu wywołania MQINQ.

TriggerInterval (MQLONG)

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to znaczenie tylko wtedy, gdy parametr *TriggerType* ma wartość MQTT_FIRST. W tym przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawi się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach dodatkowy komunikat wyzwalający może zostać wygenerowany za pomocą wywołania MQTT_FIRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekundy.

Więcej informacji na temat wyzwalania zawiera sekcja [Wyzwalanie kanałów](#).

Wartość jest nie mniejsza niż 0 i nie większa niż 999 999 999. Wartość domyślna to 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_INTERVAL przy użyciu wywołania MQINQ.

TriggerInterval (MQLONG)

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to znaczenie tylko wtedy, gdy parametr *TriggerType* ma wartość MQTT_FIRST. W tym przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawi się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach dodatkowy komunikat wyzwalający może zostać wygenerowany za pomocą wywołania MQTT_FIRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekundy.

Więcej informacji na temat wyzwalania zawiera sekcja [Wyzwalanie kanałów](#).

Wartość jest nie mniejsza niż 0 i nie większa niż 999 999 999. Wartość domyślna to 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_INTERVAL przy użyciu wywołania MQINQ.

Wersja (MQCFST)

Jest to wersja kodu produktu IBM MQ jako VVRRMMFF, gdzie:

Wersja VV

RR-wydanie

MM-poziom konserwacyjny

FF-poziom poprawek

XrCapability(MQLONG)

Określa, czy komendy produktu IBM MQ Telemetry są obsługiwane przez menedżer kolejek.

Wartość ta jest jedną z następujących wartości:

MQCAP_SUPPORTED

Obsługiwane są komponenty produktu IBM MQ Telemetry i komendy telemetryczne.

MQCAP_NOT_SUPPORTED

Komponent IBM MQ Telemetry nie został zainstalowany.

Ten atrybut jest obsługiwany tylko w systemach IBM i, Unix i Windows.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_XR_CAPABILITY przy użyciu wywołania MQINQ.

Atrybuty dla kolejek

Istnieje pięć typów definicji kolejek. Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek. Inne atrybuty kolejki mają zastosowanie tylko do określonych typów kolejek.

Typy kolejek

Menedżer kolejek obsługuje następujące typy definicji kolejek:

Kolejka lokalna

Istnieje możliwość zapisywania komunikatów w kolejce lokalnej.

z/OS W systemie z/OS można utworzyć kolejkę współużytkowaną lub prywatną.

Kolejka jest znana w programie jako *lokalna*, jeśli jej właścicielem jest menedżer kolejek, z którym połączony jest program. Komunikaty można pobierać z kolejek lokalnych oraz umieszczać je w nich.

Obiekt definicji kolejki przechowuje informacje o definicji kolejki, a także komunikaty fizyczne umieszczone w kolejce.

Kolejka lokalnego menedżera kolejek

Kolejka istnieje w menedżerze kolejek lokalnych.

z/OS Kolejka jest znana jako kolejka prywatna w systemie z/OS.

z/OS **Kolejka współużytkowana (tylko z/OS)**

Kolejka istnieje we współużytkowanym repozytorium, które jest dostępne dla wszystkich menedżerów kolejek należących do grupy współużytkowania kolejek, która jest właścicielem współużytkowanego repozytorium.

Aplikacje połączone z dowolnym menedżerem kolejek w grupie współużytkowania kolejki mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Takie kolejki są efektywnie takie same, jak kolejki lokalne. Wartością atrybutu kolejki **QType** jest MQQT_LOCAL.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty i usuwać komunikaty z kolejek tego typu. Wartością atrybutu kolejki **QType** jest MQQT_LOCAL.

Kolejka klastra

Istnieje możliwość zapisywania komunikatów w kolejce klastra w menedżerze kolejek, w którym jest ona zdefiniowana. Kolejka klastra to kolejka udostępniana przez menedżer kolejek klastra innym menedżerom kolejek w klastrze. Wartość atrybutu kolejki **QType** to MQQT_CLUSTER.

Definicja kolejki klastra jest ogłaszana w innych menedżerach kolejek w klastrze. Inne menedżery kolejek w klastrze mogą umieszczać komunikaty w kolejce klastra bez konieczności stosowania odpowiadającej jej definicji kolejki zdalnej. Kolejka klastra może zostać ogłoszona w więcej niż jednym klastrze przy użyciu listy nazw klastra.

Po ogłoszeniu kolejki każdy menedżer kolejek w klastrze może umieszczać w niej komunikaty. Aby umieścić komunikat, menedżer kolejek musi ustalić, w którym repozytorium pełnym znajduje się kolejka. Następnie do komunikatu w kolejce transmisji klastra dodawane są niektóre informacje o kierowaniu.

Menedżer kolejek może przechowywać komunikaty dla innych menedżerów kolejek w klastrze w wielu kolejkach transmisji. Menedżer kolejek można skonfigurować na dwa sposoby w celu przechowywania komunikatów w wielu kolejkach transmisji klastra. Jeśli dla atrybutu menedżera kolejek **DEFCLXQ** zostanie ustawiona wartość **CHANNEL**, na podstawie kolejki **SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE** dla każdego kanału nadawczego klastra zostanie automatycznie utworzona inna kolejka transmisji klastra. Jeśli opcja kolejki transmisji **CLCHNAME** zostanie ustawiona w taki sposób, aby była zgodna z co najmniej jednym kanałem nadawczym klastra, menedżer kolejek może przechowywać komunikaty dla zgodnych kanałów w kolejce transmisji.



Ostrzeżenie: Jeśli dedykowany produkt **SYSTEM.CLUSTER.TRANSMIT.QUEUES** jest używany z menedżerem kolejek, który został zaktualizowany z wersji produktu wcześniejszej niż IBM WebSphere MQ 7.5, należy upewnić się, że dla **SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE** opcję **SHARE/NOSHARE** ustawiono na wartość **SHARE**.

z/OS Kolejka klastra może być kolejką współużytkowaną przez członków grupy współużytkowania kolejki w programie IBM MQ for z/OS.

Kolejka zdalna

Kolejka zdalna nie jest kolejką fizyczną. Jest to lokalna definicja kolejki, która istnieje w zdalnym menedżerze kolejek. Lokalna definicja kolejki zdalnej zawiera informacje, które informują menedżera kolejek lokalnych, w jaki sposób kierować komunikaty do zdalnego menedżera kolejek.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu. Komunikaty te są umieszczane w lokalnej kolejce transmisji używanej do kierowania komunikatów do zdalnego menedżera kolejek. Aplikacje nie mogą usuwać komunikatów z kolejek zdalnych. Wartością atrybutu kolejki **QType** jest `MQQT_REMOTE`.

Można również użyć definicji kolejki zdalnej dla:

- Aliasing kolejki odpowiedzi

W tym przypadku nazwą definicji jest nazwa kolejki odpowiedzi. Więcej informacji na ten temat zawiera sekcja [Aliasy i aliasy kolejki odpowiedzi](#).

- Aliasing menedżera kolejek

W tym przypadku nazwa definicji jest aliasem dla menedżera kolejek, a nie nazwą kolejki. Więcej informacji na ten temat zawiera sekcja [Aliasy i klastry menedżera kolejek](#).

Kolejka aliasowa

Nie jest to kolejka fizyczna. Jest to nazwa alternatywna dla kolejki lokalnej, współużytkowanej kolejki, kolejki klastra lub kolejki zdalnej. Nazwa kolejki, do której jest tłumaczona alias, jest częścią definicji kolejki aliasowej.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu; komunikaty są umieszczane w kolejce, do której alias jest tłumaczący. Aplikacje mogą usuwać komunikaty z kolejek tego typu, jeśli alias jest tłumaczony na kolejkę lokalną, kolejkę współużytkowaną lub kolejkę klastra, która ma instancję lokalną. Wartością atrybutu kolejki **QType** jest `MQQT_ALIAS`.

Kolejka modelowa

Nie jest to kolejka fizyczna. Jest to zestaw atrybutów kolejki, z których można utworzyć kolejkę lokalną.

Komunikaty nie mogą być przechowywane w kolejkach tego typu.

limity kolejek

V 8.0.0.15

W przypadku produktu Wersja 8.0.0, pakiet poprawek 15 menedżer kolejek domyślnie ogranicza maksymalną wielkość pliku kolejki do 2 TB.

Kolejka - atrybuty

Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek. Inne atrybuty kolejki mają zastosowanie tylko do określonych typów kolejek. Typy kolejek, których dotyczy atrybut, są wyświetlane w [Tabela 112 na stronie 848](#) i kolejnych tabelach.

[Tabela 112 na stronie 848](#) podsumowuje atrybuty, które są specyficzne dla kolejek. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy atrybutów wyświetlane w tej sekcji to nazwy opisowe używane w wywołaniach `MQINQ` i `MQSET`; nazwy są takie same, jak w przypadku komend PCF. Jeśli komendy `MQSC` są używane do definiowania, zmieniania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Szczegółowe informacje zawiera sekcja [Komendy skryptowe \(MQSC\)](#).

Tabela 112. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli wszystkie inne atrybuty zostaną zapytane, wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068).

Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

Jeśli kolejka klastra jest otwarta tylko do zapytania lub do uzyskiwania informacji i danych wyjściowych, a także określa podstawową nazwę menedżera kolejek, to zamiast niej stosowana jest kolumna dla kolejek lokalnych.

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
AlterationDate	Data ostatniej zmiany definicji	✓		✓	✓	
AlterationTime	Czas ostatniej zmiany definicji	✓		✓	✓	
BackoutRequeueQName	Nadmierna nazwa kolejki wycofanych komunikatów	✓	✓			
BackoutThreshold	Próg wycofania	✓	✓			
BaseQName	Nazwa kolejki, do której alias jest tłumaczący			✓		
CFStrucName	Nazwa struktury narzędzia CF	✓	✓			
CLCHNAME	Nazwy kanałów nadawczych klastra	✓	✓			
ClusterName	Nazwa klastra, do którego należy kolejka	✓		✓	✓	✓
ClusterNameList	Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka	✓		✓	✓	
CLWLQueuePriority	Priorytet kolejki obciążenia klastra	✓		✓	✓	✓
CLWLQueueRank	Ranga kolejki obciążenia klastra	✓		✓	✓	✓
CLWLUseQ	Użyj kolejki zdalnej	✓				
CreationDate	Data utworzenia kolejki	✓				
CreationTime	Czas utworzenia kolejki	✓				
CurrentQDepth	Bieżąca głębokość kolejki	✓				
DefaultPutResponse	Operacja put - domyślna odp.	✓	✓	✓	✓	
DefBind	Domyślne łączenie	✓		✓	✓	✓
DefinitionType attribute	Typ definicji kolejki.	✓	✓			
DefInputOpenOption	Domyślna opcja otwarcia wejścia	✓	✓			
DefPersistence	Domyślna trwałość komunikatu	✓	✓	✓	✓	✓
DefPriority	Domyślny priorytet komunikatu	✓	✓	✓	✓	✓
DefReadAhead	Domyślny odczyt z wyprzedzeniem	✓	✓	✓		
DistLists	Obsługa listy dystrybucyjnej	✓	✓			
HardenGetBackout	Czy zachować dokładną liczbę wycofań	✓	✓			
IndexType	Typ indeksu	✓	✓			

Tabela 112. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli wszystkie inne atrybuty zostaną zapytane, wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068).

Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

Jeśli kolejka klastra jest otwarta tylko do zapytania lub do uzyskiwania informacji i danych wyjściowych, a także określa podstawową nazwę menedżera kolejek, to zamiast niej stosowana jest kolumna dla kolejek lokalnych.

(kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
InhibitGet	Określa, czy operacje pobierania dla kolejki są dozwolone	✓	✓	✓		
InhibitPut	Określa, czy dozwolone są operacje put dla kolejki	✓	✓	✓	✓	✓
InitiationQName	Nazwa kolejki inicjuj.	✓	✓			
MaxMsgLength	Maksymalna długość komunikatu w bajtach	✓	✓			
MaxQDepth	Maksymalna głębokość kolejki	✓	✓			
MsgDeliverySequence attribute	Kolejność dostarczania komunikatów	✓	✓			
NonPersistentMessage Class	Cel niezawodności dla komunikatów nietrwałych	✓	✓			
OpenInputCount	Liczba operacji otwierania dla danych wejściowych	✓				
OpenOutputCount	Liczba operacji otwierania dla danych wyjściowych	✓				
PropertyControl	Sterowanie właściwościami	✓	✓	✓		
ProcessName	Nazwa procesu	✓	✓			
QDepthHighEvent attribute	Informacja o tym, czy generowane są zdarzenia zapełnienia kolejki	✓	✓			
QDepthHighLimit	Górny limit głębokości kolejki	✓	✓			
QDepthLowEvent attribute	Informacja o tym, czy generowane są zdarzenia zapełnienia kolejki	✓	✓			
QDepthLowLimit attribute	Niski limit głębokości kolejki	✓	✓			
QDepthMaxEvent	Określa, czy generowane są zdarzenia zapełnienia kolejki	✓	✓			
QDesc	Opis kolejki	✓	✓	✓	✓	✓
QName	Nazwa kolejki	✓		✓	✓	✓
QServiceInterval	Cel dla przedziału czasu usługi kolejki	✓	✓			
QServiceIntervalEvent attribute	Określa, czy generowane są zdarzenia OK Odstęp czasu usługi lub Przedział czasu usługi OK	✓	✓			
QSGDisp attribute	Dyspozycja grupy współużytkowania kolejki	✓		✓	✓	
QueueAccounting	Gromadzenie danych rozliczeniowych w kolejce	✓	✓	✓	✓	✓
QueueMonitoring	Dane monitorowania w trybie z połączeniem dla kolejek	✓	✓			

Tabela 112. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli wszystkie inne atrybuty zostaną zapytane, wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068).

Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

Jeśli kolejka klastra jest otwarta tylko do zapytania lub do uzyskiwania informacji i danych wyjściowych, a także określa podstawową nazwę menedżera kolejek, to zamiast niej stosowana jest kolumna dla kolejek lokalnych.

(kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
QueueStatistics	Gromadzenie danych statystycznych dla kolejki	✓	✓	✓	✓	✓
QType	Typ kolejki	✓		✓	✓	✓
RemoteQMGrName	Nazwa zdalnego menedżera kolejek				✓	
RemoteQName	Nazwa kolejki zdalnej				✓	
RetentionInterval	Interwał przechowywania	✓	✓			
Scope	Określa, czy pozycja dla kolejki istnieje również w katalogu komórki.	✓		✓	✓	
Shareability	Współużytkowność kolejki	✓	✓			
StorageClass	Klasa pamięci masowej dla kolejki	✓	✓			
TriggerControl	Kontrola wyzwalacza	✓	✓			
TriggerData	Dane wyzwalacza	✓	✓			
TriggerDepth	Wyzwalacz uruchamiany zapętnieniem	✓	✓			
TriggerMsgPriority	Próg priorytetu komunikatu dla wyzwalacza.	✓	✓			
TriggerType	Typ wyzwalacza	✓	✓			
Usage attribute	Wykorzystanie kolejki	✓	✓			
XmitQName	Nazwa kolejki transmisji				✓	

Informacje pokrewne

[Kolejki klastra](#)

[Kolejki lokalne](#)

AlterationDate (MQCHAR12)

Data ostatniej zmiany definicji.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami na końcu, aby długość 12 bajtów (na przykład 1992-09-23-- , gdzie -- oznacza dwa puste znaki).

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) są zmieniane w miarę działania menedżera kolejek. Zmiany w tych atrybutach nie mają wpływu na *AlterationDate*.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Czas ostatniej zmiany definicji.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	

Jest to czas ostatniej zmiany definicji. Format czasu to HH.MM.SS, przy użyciu zegara 24-godzinnego, z zerowym zerem, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20).

- W systemie z/OS czas to GMT (Greenwich Mean Time), z zastrzeżeniem, że zegar systemowy jest ustawiony dokładnie na czas GMT.
- W innych środowiskach czas lokalny jest czasem lokalnym.

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) są zmieniane w miarę działania menedżera kolejek. Zmiany w tych atrybutach nie mają wpływu na *AlterationTime*.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_TIME_LENGTH.

Nazwa QName BackoutRequeue(MQCHAR48)

Jest to nadmierna nazwa kolejki wycofanych komunikatów. Oprócz tego, że można wykonać zapytanie o jego wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość tego atrybutu.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Aplikacje działające w produkcie WebSphere Application Server i te, które korzystają z narzędzi serwera aplikacji produktu IBM MQ, używają tego atrybutu do określenia, gdzie powinny być wyświetlane komunikaty, których kopie zapasowe zostały wycofane. W przypadku wszystkich innych aplikacji menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość atrybutu.

Produkt IBM MQ classes for JMS używa tego atrybutu do określenia, gdzie należy przestać komunikat, dla którego została już utworzona kopia zapasowa, maksymalnej liczby razy określonej w atrybucie *BackoutThreshold*.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_BACKOUT_REQ_Q_NAME w wywołaniu MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)

Jest to próg wycofania. Oprócz tego, że można wykonać zapytanie o jego wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość atrybutu.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Aplikacje działające w produkcie WebSphere Application Server i te, które korzystają z narzędzi serwera aplikacji produktu IBM MQ, będą używać tego atrybutu w celu określenia, czy należy utworzyć kopię zapasową komunikatu. W przypadku wszystkich innych aplikacji menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość atrybutu.

Produkt IBM MQ classes for JMS używa tego atrybutu do określenia, ile razy komunikat ma być wycofany przed przestaniem komunikatu do kolejki określonej atrybutem *BackoutRequeueQName*.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_BACKOUT_THRESHOLD przy użyciu wywołania MQINQ.

BaseQName (MQCHAR48)

Jest to nazwa kolejki, która jest zdefiniowana dla lokalnego menedżera kolejek.

Lokalna	Model	Alias	Zdalny	Klaster
		X		

(Więcej informacji na temat nazw kolejek znajduje się w sekcji [MQOD- ObjectName](#)). Kolejka jest jednym z następujących typów:

MQQT_LOCAL

Kolejka lokalna.

MQQT_REMOTE

Lokalna definicja kolejki zdalnej.

MQQT_CLUSTER

Kolejka klastra.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_BASE_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

BaseType (MQCFIN)

Typ obiektu, do którego alias jest rozstrzygany.

Lokalna	Model	Alias	Zdalny	Klaster
		X		

Jest to jedna z następujących wartości:

Kolejka MQOT_Q

Podstawowy typ obiektu to kolejka

MQOT_TOPIC

Podstawowy typ obiektu to temat

CFStrucName (MQCHAR12)

Jest to nazwa struktury narzędzia CF, w której zapisywane są komunikaty w kolejce. Pierwszy znak nazwy mieści się w zakresie od A do Z, a pozostałe znaki są w zakresie od A do Z, od 0 do 9 lub puste.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Aby uzyskać pełną nazwę struktury w narzędziu CF, przyrostek wartości atrybutu menedżera kolejek produktu QSGName z wartością atrybutu kolejki produktu CFStrucName .

Ten atrybut ma zastosowanie tylko do współużytkowanych kolejek. Jest on ignorowany, jeśli wartość QSGDisp nie ma wartości MQQSGD_SHARED.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CF_STRUC_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez MQ_CF_STRUC_NAME_LENGTH.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

ClusterChannelNazwa (MQCHAR20)

ClusterChannelNazwa to ogólna nazwa kanałów nadawczych klastra, które używają tej kolejki jako kolejki transmisji. Atrybut określa, które kanały nadawcze klastra wysyłają komunikaty do kanału odbiorczego klastra z tej kolejki transmisji klastra.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Menedżer kolejek jest domyślnie skonfigurowany w taki sposób, aby wszystkie kanały nadawcze klastra wysyłały komunikaty z pojedynczej kolejki transmisji: `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Konfigurację domyślną można zmienić, modyfikując atrybut **DefClusterXmitQueueType** menedżera kolejek. Wartością domyślną tego atrybutu jest `SCTQ`. Wartość tę można zmienić na `CHANNEL`. Jeśli atrybut **DefClusterXmitQueueType** zostanie ustawiony na wartość `CHANNEL`, dla każdego kanału nadawczego klastra domyślnie zostanie użyta konkretna kolejka transmisji klastra `SYSTEM.CLUSTER.TRANSMIT.ChannelName`.

Kanał nadawczy klastra dla atrybutu `ClusterChannelName` kolejki transmisji można również ustawić ręcznie. Komunikaty przeznaczone dla menedżera kolejek połączonego kanałem nadawczym klastra są przechowywane w kolejce transmisji identyfikującej kanał nadawczy klastra. Nie są one przechowywane w domyślnej kolejce transmisji klastra. Jeśli dla atrybutu `ClusterChannelName` zostaną ustawione wartości puste, po zrestartowaniu kanału zostanie on przetaczony na domyślną kolejkę transmisji klastra. Kolejka domyślna to `SYSTEM.CLUSTER.TRANSMIT.ChannelName` lub `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, w zależności od wartości atrybutu `DefClusterXmitQueueType` menedżera kolejek.

Określenie w atrybucie **ClusterChannelName** gwiazdek ("*") umożliwia powiązanie kolejki transmisji z zestawem kanałów nadawczych klastra. Gwiazdki mogą znajdować się na początku, na końcu lub na dowolnej liczbie miejsc w środku łańcucha nazwy kanału. Długość atrybutu **ClusterChannelName** jest ograniczona do 20 znaków: `MQ_CHANNEL_NAME_LENGTH`.

ClusterName (MQCHAR48)

Jest to nazwa klastra, do którego należy kolejka.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Jeśli kolejka należy do więcej niż jednego klastra, `ClusterNameList` określa nazwę obiektu listy nazw, która identyfikuje klastry, a `ClusterName` jest pusta. Co najmniej jedna z wartości `ClusterName` i `ClusterNameList` musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora `MQCA_CLUSTER_NAME` przy użyciu wywołania `MQINQ`. Długość tego atrybutu jest podana przez wartość `MQ_CLUSTER_NAME_LENGTH`.

ClusterNameList (MQCHAR48)

Jest to nazwa obiektu listy nazw, który zawiera nazwy klastrów, do których należy ta kolejka.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	

Jeśli kolejka należy do tylko jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć opcji `ClusterName` do określenia nazwy klastra, w którym to przypadku pole `ClusterNameList` jest puste. Co najmniej jedna z wartości `ClusterName` i `ClusterNameList` musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora `MQCA_CLUSTER_NAMELIST` z wywołaniem `MQINQ`. Długość tego atrybutu jest podana przez wartość `MQ_NAMELIST_NAME_LENGTH`.

CLWLQueuePriority (MQLONG)

Jest to priorytet kolejki obciążenia klastra, wartość z zakresu od 0 do 9, która reprezentuje priorytet kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Więcej informacji na ten temat zawiera sekcja [Kolejki klastrów](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_Q_PRIORITY z wywołaniem MQINQ.

CLWLQueueRank (MQLONG)

Jest to ranga kolejki obciążenia klastra, wartość z zakresu od 0 do 9, reprezentująca rangę kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Więcej informacji na ten temat zawiera sekcja [Kolejki klastrów](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_Q_RANK przy użyciu wywołania MQINQ.

CLWLUseQ (MQLONG)

Definiuje zachowanie operacji MQPUT w przypadku, gdy kolejka docelowa ma zarówno instancję lokalną, jak i co najmniej jedną zdalną instancję klastra. Jeśli operacja put pochodzi z kanału klastra, ten atrybut nie ma zastosowania.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Wartość ta jest jedną z następujących wartości:

MQCLWL_USEQ_ANY

Użyj kolejek zdalnych i lokalnych.

MQCLWL_USEQ_LOCAL

Nie należy używać kolejek zdalnych.

MQCLWL_USEQ_AS_Q_MGR

Dziedzicz definicję z menedżera kolejek MQIA_CLWL_USEQ.

Więcej informacji na ten temat zawiera sekcja [Kolejki klastrów](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLWL_USEQ z wywołaniem MQINQ. Długość tego atrybutu jest podana przez MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)

Data utworzenia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów (na przykład 2013-09-23--), gdzie -- oznacza 2 puste znaki).

- W systemie IBM data utworzenia kolejki może różnić się od daty bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CREATION_DATE przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)

Jest to czas utworzenia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Format czasu to HH.MM.SS, przy użyciu zegara 24-godzinnego, z zerowym zerem, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20).

- W systemie z/OS czas to GMT (Greenwich Mean Time), z zastrzeżeniem, że zegar systemowy jest ustawiony dokładnie na czas GMT.
- W innych środowiskach czas lokalny jest czasem lokalnym.
- W systemie IBM czas utworzenia kolejki może różnić się od czasu utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CREATION_TIME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)

To jest liczba komunikatów znajdujących się aktualnie w kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Wartość ta jest zwiększana podczas wywołania MQPUT i podczas wycofywania wywołania MQGET. Jest ono zmniejszane podczas wywołania MQGET bez przeglądania i podczas wycofywania wywołania MQPUT. Wynika z tego, że liczba obejmuje komunikaty, które zostały umieszczone w kolejce w ramach jednostki pracy, ale nie zostały jeszcze zatwierdzone, nawet jeśli nie są zakwalifikowane do pobrania za pomocą wywołania MQGET. Podobnie, nie obejmuje ona komunikatów, które zostały pobrane w ramach jednostki pracy przy użyciu wywołania MQGET, ale które nie zostały jeszcze zatwierdzone.

Liczba ta obejmuje również komunikaty, które przeszły czas utraty ważności, ale nie zostały jeszcze odrzucone, mimo że te komunikaty nie są zakwalifikowane do pobrania. Więcej informacji na ten temat zawiera sekcja [MQMD-Expiry field](#).

Przetwarzanie jednostkowe i segmentacja komunikatów może spowodować, że *CurrentQDepth* przekroczy *MaxQDepth*. Nie ma to jednak wpływu na pobieranie komunikatów; wszystkie komunikaty znajdujące się w kolejce mogą być pobierane za pomocą wywołania MQGET w normalny sposób.

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CURRENT_Q_DEPTH przy użyciu wywołania MQINQ.

Odpowiedź DefaultPut(MQLONG)

Określa typ odpowiedzi, która ma być używana na potrzeby operacji put dla kolejki, gdy aplikacja określa wartość MQPMO_RESPONSE_AS_Q_DEF.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	

Jest to jedna z następujących wartości:

MQPRT_SYNC_RESPONSE

Operacja put jest wykonywana synchronicznie, zwracając odpowiedź.

MQPRT_ASYNC_RESPONSE

Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

DefBind (MQLONG)

Jest to powiązanie domyślne, które jest używane, gdy w wywołaniu MQOPEN określono parametr MQOO_BIND_AS_Q_DEF, a kolejka jest kolejką klastra.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Wartość ta jest jedną z następujących wartości:

MQBND_BIND_ON_OPEN

Powiązanie ustalone przez wywołanie MQOPEN.

MQBND_BIND_NOT_FIXED

Powiązanie nie zostało ustalone.

MQBND_BIND_ON_GROUP

Umożliwia aplikacji żądanie, aby grupa komunikatów była przydzielona do tej samej instancji docelowej. Ponieważ ta wartość jest nowa w produkcie IBM WebSphere MQ 7.1, nie może być używana, jeśli dowolna z aplikacji otwierających tę kolejkę łączy się z programem IBM WebSphere MQ 7.0.1 lub wcześniejszymi menedżerami kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_BIND przy użyciu wywołania MQINQ.

DefinitionType (MQLONG)

Wskazuje, w jaki sposób została zdefiniowana kolejka.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Wartość ta jest jedną z następujących wartości:

MQQDT_PREDEFINIOWANE

Kolejka jest stałą kolejką utworzoną przez administratora systemu. Tylko administrator systemu może go usunąć.

Predefiniowane kolejki są tworzone za pomocą komendy MQSC DEFINE i mogą być usuwane tylko za pomocą komendy MQSC DELETE. Predefiniowanych kolejek nie można tworzyć z kolejek modelowych.

Komendy mogą być wydawane przez operatora lub przez autoryzowanego użytkownika wysyłającego komunikat komendy do kolejki wejściowej komend (więcej informacji na ten temat zawiera sekcja [CommandInputQName attribute](#)).

MQQDT_PERMANENT_DYNAMIC

Kolejka to kolejka trwała, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. W definicji kolejki modelowej wartość MQQDT_PERMANENT_DYNAMIC jest określona dla atrybutu *DefinitionType*.

Ten typ kolejki można usunąć przy użyciu wywołania MQCLOSE. Więcej szczegółów na ten temat zawiera sekcja [“MQCLOSE-zamknięcie obiektu”](#) na stronie 655.

Wartością atrybutu *QSGDIsip* dla trwałej kolejki dynamicznej jest MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC

Kolejka jest kolejką tymczasową, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. W definicji kolejki modelowej wartość MQQDT_TEMPORARY_DYNAMIC jest określona dla atrybutu *DefinitionType*.

Ten typ kolejki jest automatycznie usuwany przez wywołanie MQCLOSE, gdy jest on zamykany przez aplikację, która go utworzyła.

Wartością atrybutu *QSGDsp* dla tymczasowej kolejki dynamicznej jest MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC

Kolejka jest współużytkowaną statą kolejką, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. Definicja kolejki modelowej ma wartość MQQDT_SHARED_DYNAMIC dla atrybutu *DefinitionType*.

Ten typ kolejki można usunąć przy użyciu wywołania MQCLOSE. Więcej szczegółów na ten temat zawiera sekcja [“MQCLOSE-zamknięcie obiektu”](#) na stronie 655.

Wartością atrybutu *QSGDsp* dla współużytkowanej kolejki dynamicznej jest MQQSGD_SHARED.

Ten atrybut w definicji kolejki modelowej nie wskazuje, w jaki sposób została zdefiniowana kolejka modelowa, ponieważ kolejki modelowe są zawsze predefiniowane. Zamiast tego wartość tego atrybutu w kolejce modelowej jest używana do określenia *DefinitionType* każdej kolejki dynamicznej utworzonej z definicji kolejki modelowej przy użyciu wywołania MQOPEN.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEFINITION_TYPE z wywołaniem MQINQ.

DefInputOpenOption (MQLONG)

Jest to domyślny sposób otwierania kolejki na potrzeby wprowadzania danych.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Ma zastosowanie, jeśli podczas otwierania kolejki opcja MQOO_INPUT_AS_Q_DEF została określona w wywołaniu MQOPEN. Wartość ta jest jedną z następujących wartości:

MQOO_INPUT_EXCLUSIVE

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie nie powiodło się z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla danych wejściowych dowolnego typu (MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest aktualnie otwarta przez tę lub inną aplikację z opcją MQOO_INPUT_SHARED, ale kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta z opcją MQOO_INPUT_EXCLUSIVE.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_INPUT_OPEN_OPTION w wywołaniu MQINQ.

DefPersistence (MQLONG)

Jest to domyślna trwałość komunikatów w kolejce. Ma zastosowanie, jeśli komunikat MQPER_PERSISTENCE_AS_Q_DEF został określony w deskrytorze komunikatu w momencie umieszczania komunikatu.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w definicji *pierwszej* w ścieżce w czasie wywołania MQPUT lub MQPUT1. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Wartość ta jest jedną z następujących wartości:

MQPER_PERSISTENT

Komunikat zachować awarię systemu i restarty menedżera kolejek. Komunikaty trwałe nie mogą być umieszczane w następujących systemach:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, które są odwzorowywać na obiekt CFSTRUCT na poziomie CFLEVEL (2) lub poniżej, lub gdzie obiekt CFSTRUCT jest zdefiniowany jako RECOVER (NO).

Komunikaty trwałe mogą być umieszczane w statycznych kolejkach dynamicznych i predefiniowanych kolejkach.

MQPER_NOT_PERSISTENT

Zazwyczaj komunikat nie jest w stanie przetrwać awariom systemu lub restartami menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartu menedżera kolejek na pamięci dyskowej zostanie znaleziona nienaruszona kopia komunikatu.

W przypadku kolejek współużytkowanych komunikaty nietrwałe *do* przeżywają restarty menedżerów kolejek w grupie współużytkowania kolejki, ale nie przeżywają niepowodzeń narzędzia CF używanego do przechowywania komunikatów w kolejkach współużytkowanych.

Zarówno komunikaty trwałe, jak i nietrwałe mogą istnieć w tej samej kolejce.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PERSISTENCE z wywołaniem MQINQ.

DefPriority (MQLONG)

Jest to domyślny priorytet komunikatów w kolejce. Ma to zastosowanie, jeśli wartość MQPRI_PRIORITY_AS_Q_DEF została określona w deskrypcji komunikatu, gdy komunikat jest umieszczany w kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek istnieje więcej niż jedna definicja, domyślny priorytet komunikatu jest przyjmowany z wartości tego atrybutu w definicji *pierwszej* podanej w ścieżce w czasie operacji put. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Sposób, w jaki komunikat jest umieszczany w kolejce, zależy od wartości atrybutu *MsgDeliverySequence* kolejki:

- Jeśli atrybut *MsgDeliverySequence* ma wartość MQMDS_PRIORITY, to położenie logiczne, w którym komunikat jest umieszczany w kolejce, zależy od wartości pola *Priority* w deskrypcji komunikatu.
- Jeśli atrybut *MsgDeliverySequence* ma wartość MQMDS_FIFO, komunikaty są umieszczane w kolejce tak, jakby miały priorytet równy *DefPriority* rozstrzygniętej kolejki, niezależnie od wartości

pola *Priority* w deskrytorze komunikatu. Jednak pole *Priority* zachowuje wartość określoną przez aplikację, która wstawiła komunikat. Więcej informacji na ten temat zawiera sekcja Atrybut kolejnościMsgDelivery.

Priorytety są w zakresie od zera (najniższy) do *MaxPriority* (najwyższy); patrz atrybutMaxPriority.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PRIORITY w wywołaniu MQINQ.

DefReadAhead (MQLONG)

Określa domyślne zachowanie odczytu z wyprzedzeniem dla nietrwałych komunikatów dostarczanych do klienta.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X		

DefReadAhead można ustawić na jedną z następujących wartości:

MQREADA_NO

Komunikaty nietrwałe nie są wysyłane z wyprzedzeniem do klienta przed ich żądaniami. Jeśli działanie klienta zostanie zakończone nieprawidłowo, może zostać utracony maksymalnie jeden komunikat nietrwały.

MQREADA_YES

Komunikaty nietrwałe są wysyłane z wyprzedzeniem do klienta, zanim aplikacja je zażąda. Komunikaty nietrwałe mogą zostać utracone, jeśli klient zakończy się nieprawidłowo lub jeśli klient nie zużywa wszystkich wysłanych wiadomości.

MQREADA_DISABLED

Odczyt z wyprzedzeniem dla nietrwałych komunikatów, które nie zostały włączone dla tej kolejki. Komunikaty nie są wysyłane z wyprzedzeniem do klienta niezależnie od tego, czy aplikacja kliencka żąda odczytu z wyprzedzeniem.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_READ_AHEAD z wywołaniem MQINQ.

DefPResp (MQLONG)

Domyślny atrybut typu put odpowiedzi (put response type-DEFPRESP) definiuje wartość używaną przez aplikację, gdy typ PutResponsew produkcie MQPMO został ustawiony na wartość MQPMO_RESPONSE_AS_Q_DEF. Ten atrybut jest poprawny dla wszystkich typów kolejek.

Lokalna	Model	Alias	Zdalny	Klaster
Tak	Tak	Tak	Tak	Tak

Wartość ta jest jedną z następujących wartości:

SYNCHRONICZNY

Operacja put jest wydawana synchronicznie, zwracając odpowiedź.

ASYNCHRONICZNY

Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PUT_RESPONSE_TYPE z wywołaniem MQINQ.

DistLists (MQLONG)

Wskazuje, czy komunikaty listy dystrybucyjnej mogą być umieszczane w kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Agent kanału komunikatów (MCA) ustawia atrybut w celu poinformowania lokalnego menedżera kolejek, czy menedżer kolejek na drugim końcu kanału obsługuje listy dystrybucyjne. Ten ostatni menedżer kolejek (nazywany menedżerem kolejek *partnering*) jest tym, który następnie odbiera komunikat po usunięciu go z lokalnej kolejki transmisji przez wysyłający agent MCA.

Wysyłający agent MCA ustawia atrybut za każdym razem, gdy nawiązuje połączenie z odbierającym MCA w partnerskim menedżerze kolejek. W ten sposób wysyłający agent MCA może spowodować, że lokalny menedżer kolejek umieje w kolejce transmisji tylko komunikaty, które może poprawnie przetworzyć partnerski menedżer kolejek.

Ten atrybut jest przeznaczony przede wszystkim do użycia z kolejkami transmisji, ale opisane przetwarzanie jest wykonywane niezależnie od użycia zdefiniowanego dla kolejki (patrz sekcja Atrybut użycia).

Wartość ta jest jedną z następujących wartości:

MQDL_SUPPORTED

Komunikaty listy dystrybucyjnej mogą być zapisywane w kolejce i przekazywane do partnerskiego menedżera kolejek w tej postaci. Zmniejsza to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych.

MQDL_NOT_SUPPORTED

Komunikaty listy dystrybucyjnej nie mogą być przechowywane w kolejce, ponieważ partnerski menedżer kolejek nie obsługuje list dystrybucyjnych. Jeśli aplikacja umieszcza komunikat z listą dystrybucyjną i ten komunikat ma zostać umieszczony w tej kolejce, menedżer kolejek rozdziela komunikat listy dystrybucyjnej i umieszcza poszczególne komunikaty w kolejce zamiast tego komunikatu. Zwiększa to ilość przetwarzania wymaganą do wysłania komunikatu do wielu miejsc docelowych, ale zapewnia, że komunikaty są przetwarzane poprawnie przez partnerski menedżer kolejek.

Aby określić wartość atrybutu, należy użyć selektora MQIA_DIST_LISTS z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Ten atrybut nie jest obsługiwany w systemie z/OS.

HardenGetBackout (MQLONG)

Dla każdego komunikatu jest zachowana liczba określająca, ile razy komunikat jest pobierany przez wywołanie MQGET w ramach jednostki pracy, a następnie ta jednostka pracy została wycofana.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Liczba ta jest dostępna w polu *BackoutCount* w deskrypcji komunikatu po zakończeniu wywołania MQGET.

Licznik wycofań komunikatów jest restartowany od restartów menedżera kolejek. Jednak aby upewnić się, że liczba jest dokładna, informacje muszą być *utwardzone* (rejestrowane na dysku lub innym stałym urządzeniu pamięci masowej) za każdym razem, gdy wywołanie MQGET pobiera komunikat w ramach jednostki pracy dla tej kolejki. Jeśli ta opcja nie zostanie wykonana, menedżer kolejek nie powiedzie się, a wywołania MQGET są wycofane, licznik może lub nie może być zwiększony.

Utwardzanie informacji dla każdego wywołania MQGET w ramach jednostki pracy nakłada jednak dodatkowe koszty przetwarzania, dlatego atrybut *HardenGetBackout* należy ustawić na wartość MQQA_BACKOUT_HARTOWANE tylko wtedy, gdy jest to istotne, aby liczba była dokładna.

W systemach IBM i, Systemy UNIX i Windows liczba wycofanych komunikatów jest zawsze hartowana, niezależnie od ustawienia tego atrybutu.

Dozwolone są następujące wartości:

MQQA_BACKOUT_HARTOWANA

Hartowanie jest używane w celu zapewnienia, że liczba wycofań komunikatów w tej kolejce jest dokładna.

MQQA_BACKOUT_NOT_HARTOWANE

Utwardzanie nie jest używane, aby upewnić się, że liczba wycofań komunikatów w tej kolejce jest dokładna. W związku z tym liczba ta może być mniejsza niż powinna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_HARDEN_GET_BACKOUT przy użyciu wywołania MQINQ.

IndexType (MQLONG)

Określa typ indeksu, który menedżer kolejek przechowuje dla komunikatów w kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Typ wymaganego indeksu zależy od sposobu pobierania komunikatów przez aplikację oraz od tego, czy kolejka jest kolejką współużytkowaną, czy niewspółużytkowaną (patrz [Atrybut QSGDisp](#)). Dla produktu *IndexType* możliwe są następujące wartości:

MQIT_NONE

Menedżer kolejek dla tej kolejki nie jest obsługiwany przez menedżer kolejek. Tej wartości należy użyć dla kolejek, które są zwykle przetwarzane sekwencyjnie, tj. bez użycia kryteriów wyboru w wywołaniu MQGET.

MQIT_MSG_ID,

Menedżer kolejek przechowuje indeks, który używa identyfikatorów komunikatów komunikatów w kolejce. Użyj tej kolejki wartości, w której aplikacja zwykle pobiera komunikaty przy użyciu identyfikatora komunikatu jako kryterium wyboru w wywołaniu MQGET.

ID_MQIT_CORREL_ID

Menedżer kolejek przechowuje indeks, który korzysta z identyfikatorów korelacji komunikatów w kolejce. Tej wartości należy użyć w przypadku kolejek, w których aplikacja zwykle pobiera komunikaty przy użyciu identyfikatora korelacji jako kryterium wyboru w wywołaniu MQGET.

MQIT_MSG_TOKEN,

Ważne: Ten typ indeksu powinien być używany tylko dla kolejek używanych z produktem IBM MQ Workflow dla produktu z/OS.

Menedżer kolejek przechowuje indeks, który używa znaczników komunikatów komunikatów w kolejce do użycia z funkcjami menedżera obciążenia (WLM) produktu z/OS.

Opcja *musi* określać tę opcję dla kolejek zarządzanych przez WLM; nie należy określać jej dla żadnego innego typu kolejki. Nie należy również używać tej wartości dla kolejki, w której aplikacja nie korzysta z funkcji menedżera obciążenia produktu z/OS, ale pobiera komunikaty przy użyciu znacznika komunikatu jako kryterium wyboru w wywołaniu MQGET.

MQIT_GROUP_ID

Menedżer kolejek przechowuje indeks, który korzysta z identyfikatorów grup komunikatów w kolejce. Ta wartość *musi* być używana dla kolejek, w których aplikacja pobiera komunikaty przy użyciu opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET.

Kolejka o tym typie indeksu nie może być kolejką transmisji. Kolejka współużytkowana z tym typem indeksu musi być zdefiniowana w celu odwzorowania na obiekt CFSTRUCT na poziomie CFLEVEL (3) lub wyższym.

Uwaga:

1. Fizyczna kolejność komunikatów w kolejce o typie indeksu MQIT_GROUP_ID nie jest zdefiniowana, ponieważ kolejka jest zoptymalizowana pod kątem wydajnego pobierania komunikatów za pomocą opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET. Oznacza to, że fizyczna kolejność komunikatów nie jest zazwyczaj kolejką, w której komunikaty dotarły do kolejki.
2. Jeśli w kolejce MQIT_GROUP_ID znajduje się *MsgDeliverySequence* o wartości MQMDS_PRIORITY, menedżer kolejek używa priorytetów komunikatów 0 i 1 w celu zoptymalizowania pobierania komunikatów w porządku logicznym. W rezultacie pierwszy

komunikat w grupie nie może mieć priorytetu zero lub jeden; jeśli tak się stanie, to komunikat jest przetwarzany tak, jakby miał priorytet równy dwóm. Pole *Priority* w strukturze MQMD nie jest zmieniane.

Więcej informacji na temat grup komunikatów można znaleźć w opisie opcji grupy i segmentu w produkcie MQGMO-pole opcji.

Typ indeksu, który powinien być używany w różnych przypadkach, jest wyświetlany w produkcie Tabela 113 na stronie 862 i w produkcie Tabela 114 na stronie 863.

<i>Tabela 113. Sugerowane lub wymagane wartości typu indeksu kolejki, jeśli nie określono parametru MQGMO_LOGICAL_ORDER</i>		
Kryteria wyboru dla wywołania MQGET	Typ indeksu dla kolejki niewspółużytkowanej	Typ indeksu dla kolejki współużytkowanej
Brak	Dowolna	Dowolna
Wybór przy użyciu jednego identyfikatora:		
Identyfikator komunikatu	Sugerowana wartość MQIT_MSG_ID	Wymagany jest obiekt MQIT_NONE lub MQIT_MSG_ID; sugerowana wartość MQIT_MSG_ID
Identyfikator korelacji	Sugerowana wartość MQIT_CORREL_ID	Wymagany jest obiekt MQIT_CORREL_ID
Identyfikator grupy	Sugerowana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu dwóch identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji	Sugerowane wartości MQIT_MSG_ID lub MQIT_CORREL_ID	Wymagane wartości MQIT_NONE lub MQIT_MSG_ID lub MQIT_CORREL_ID (W przypadku wydajności zaleca się, aby typ indeksu był zgodny z polem MQMD, które będzie miało najbardziej odrębne klucze).
Identyfikator komunikatu plus identyfikator grupy	Sugerowane wartości MQIT_MSG_ID lub MQIT_GROUP_ID	Nieobsługiwane
Identyfikator korelacji plus identyfikator grupy	Sugerowano MQIT_CORREL_ID lub MQIT_GROUP_ID	Nieobsługiwane
Wybór przy użyciu trzech identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji plus identyfikator grupy	Sugerowane parametry MQIT_MSG_ID lub MQIT_CORREL_ID lub MQIT_GROUP_ID	Nieobsługiwane
Wybór przy użyciu kryteriów dotyczących grupy:		
Identyfikator grupy plus numer kolejny komunikatu	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID

Tabela 113. Sugerowane lub wymagane wartości typu indeksu kolejki, jeśli nie określono parametru MQGMO_LOGICAL_ORDER (kontynuacja)

Kryteria wyboru dla wywołania MQGET	Typ indeksu dla kolejki niewspółużytkowanej	Typ indeksu dla kolejki współużytkowanej
Numer kolejny komunikatu (musi mieć wartość 1)	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu znacznika komunikatu:		
Znacznik komunikatu do użycia aplikacji	Nie używaj wartości MQIT_MSG_TOKEN	
Znacznik komunikatu dla użycia WLM	Wymagany znacznik MQIT_MSG_TOKEN	Nieobsługiwane

Tabela 114. Sugerowane lub wymagane wartości typu indeksu kolejki, jeśli określono parametr MQGMO_LOGICAL_ORDER

Kryteria wyboru dla wywołania MQGET	Typ indeksu dla kolejki niewspółużytkowanej	Typ indeksu dla kolejki współużytkowanej
Brak	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu jednego identyfikatora:		
Identyfikator komunikatu	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator korelacji	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu dwóch identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator komunikatu plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator korelacji plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Wybór przy użyciu trzech identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INDEX_TYPE z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

InhibitGet (MQLONG)

Określa, czy operacje pobierania dla tej kolejki są dozwolone.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X		

Jeśli kolejka jest kolejką aliasową, operacje get muszą być dozwolone zarówno dla aliasu, jak i dla kolejki podstawowej w czasie operacji pobierania, aby wywołanie MQGET powiodło się. Wartość ta jest jedną z następujących wartości:

MQQA_GET_INHIBITED

Operacje pobierania są zablokowane.

Wywołania MQGET nie powiodły się z kodem przyczyny MQRC_GET_INHIBITED. Dotyczy to wywołań MQGET, które określają parametr MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Uwaga: Jeśli wywołanie MQGET działające w ramach jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu *InhibitGet* z późniejszym wynikiem na wartość MQQA_GET_INHIBITED nie zapobiega zatwierdzającej jednostce pracy.

MQQA_GET_ALLOWED

Operacje pobierania są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_GET przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InhibitPut (MQLONG)

Określa, czy operacje put dla tej kolejki są dozwolone.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek istnieje więcej niż jedna definicja, należy zezwolić na operacje put dla *wszystkich* definicji w ścieżce (w tym definicji aliasów menedżera kolejek) w czasie operacji put, aby wywołanie MQPUT lub MQPUT1 powiodło się. Wartość ta jest jedną z następujących wartości:

MQQA_PUT_INHIBITED

Operacje put są zablokowane.

Wywołania MQPUT i MQPUT1 kończą się niepowodzeniem z kodem przyczyny MQRC_PUT_INHIBITED.

Uwaga: Jeśli wywołanie MQPUT działające w obrębie jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu *InhibitPut* w wyniku działania komendy MQQA_PUT_INHIBITED nie uniemożliwia zatwierdzenia jednostki pracy.

MQQA_PUT_ALLOWED

Operacje put są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_PUT przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InitiationQName (MQCHAR48)

Jest to nazwa kolejki zdefiniowanej w lokalnym menedżerze kolejek. Kolejka musi być typu MQQT_LOCAL.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Menedżer kolejek wysyła komunikat wyzwalacza do kolejki inicjuj, gdy uruchamianie aplikacji jest wymagane w wyniku komunikatu docierającego do kolejki, do której należy ten atrybut. Kolejka inicjująca musi być monitorowana przez aplikację monitora wyzwalacza, która uruchamia odpowiednią aplikację po odebraniu komunikatu wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_INITIATION_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

MaxMsgDługość (MQLONG)

Jest to górna granica długości najdłuższego komunikatu *fizycznego*, który może zostać umieszczony w kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Jednak ponieważ atrybut kolejki *MaxMsgLength* może być ustawiony niezależnie od atrybutu menedżera kolejek produktu *MaxMsgLength*, rzeczywisty górny limit długości najdłuższego komunikatu fizycznego, który może zostać umieszczony w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, istnieje możliwość umieszczenia komunikatu *logicznego*, który jest dłuższy niż mniejszy z dwóch atrybutów produktu *MaxMsgLength*, ale tylko wtedy, gdy aplikacja określa flagę MQMF_SEGMENTATION_ALLOWED w deskryptorach MQMD. Jeśli ta opcja jest określona, górna granica długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zwykle ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym aplikacja jest uruchomiona, powodują zmniejszenie dolnego limitu.

Próba umieszczenia w kolejce komunikatu, który jest zbyt długi, kończy się niepowodzeniem z jednym z następujących kodów przyczyny:

- MQRC_MSG_TOO_BIG_FOR_Q, jeśli komunikat jest zbyt duży dla kolejki
- MQRC_MSG_TOO_BIG_FOR_Q_MGR, jeśli komunikat jest zbyt duży dla menedżera kolejek, ale nie jest zbyt duży w przypadku kolejki

Dolny limit dla atrybutu *MaxMsgLength* ma wartość zero; górny limit wynosi 100 MB (104 857 600 bajtów).

Więcej informacji na ten temat zawiera sekcja [Parametr MQPUT- BufferLength](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_MSG_LENGTH przy użyciu wywołania MQINQ.

MaxQDepth (MQLONG)

Jest to zdefiniowany górny limit dla liczby komunikatów fizycznych, które mogą istnieć w kolejce w dowolnym momencie.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Próba umieszczenia komunikatu w kolejce, która zawiera już komunikaty produktu *MaxQDepth*, kończy się niepowodzeniem z kodem przyczyny MQRC_Q_FULL.

Przetwarzanie jednostkowe i segmentacja komunikatów może spowodować, że rzeczywista liczba komunikatów fizycznych w kolejce przekracza *MaxQDepth*. Nie ma to jednak wpływu na możliwość pobierania komunikatów; *wszystkie* komunikaty w kolejce mogą być pobierane za pomocą wywołania MQGET.

Wartość tego atrybutu jest równa zero lub większa. Górna granica jest określana przez środowisko:

- W systemach AIX, HP-UX, z/OS, Solaris, Linux i Windows wartość nie może być większa niż 999 999 999.
- W systemie IBM i wartość nie może przekroczyć 640 000.

Uwaga: Ilość miejsca w pamięci masowej dostępnego dla kolejki może zostać wyczerpana, nawet jeśli w kolejce znajduje się mniej niż *MaxQDepth* komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_Q_DEPTH przy użyciu wywołania MQINQ.

Sekwencja *MsgDelivery*(*MQLONG*)

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Określa kolejność, w jakiej wywołanie MQGET zwraca komunikaty do aplikacji:

MQMDS_FIFO

Komunikaty są zwracane w kolejności FIFO (najpierw w kolejności, w pierwszej kolejności).

Wywołanie MQGET zwraca *pierwszy* komunikat, który spełnia kryteria wyboru określone w wywołaniu, niezależnie od priorytetu komunikatu.

MQMDS_PRIORITY,

Komunikaty są zwracane w kolejności priorytetów.

Wywołanie MQGET zwraca komunikat *highest-priority*, który spełnia kryteria wyboru określone w wywołaniu. W ramach każdego poziomu priorytetu komunikaty są zwracane w kolejności FIFO (najpierw w kolejności, w pierwszej kolejności).

- W systemie z/OS, jeśli kolejka ma *IndexType* o wartości MQIT_GROUP_ID, atrybut *MsgDeliverySequence* określa kolejność, w jakiej grupy komunikatów są zwracane do aplikacji. Określona sekwencja, w której zwracane są grupy, jest określana na podstawie pozycji lub priorytetu pierwszego komunikatu w każdej grupie. Fizyczna kolejność komunikatów w kolejce nie jest zdefiniowana, ponieważ kolejka jest zoptymalizowana pod kątem wydajnego pobierania komunikatów za pomocą opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET.
- W systemie z/OS, jeśli parametr *IndexType* to MQIT_GROUP_ID, a parametr *MsgDeliverySequence* to MQMDS_PRIORITY, menedżer kolejek używa wartości zero priorytetów komunikatów i jeden w celu zoptymalizowania pobierania komunikatów w porządku logicznym. W rezultacie pierwszy komunikat w grupie nie może mieć priorytetu zero lub jeden; jeśli tak się stanie, to komunikat jest przetwarzany tak, jakby miał priorytet równy dwóm. Pole *Priority* w strukturze MQMD nie jest zmieniane.

Jeśli odpowiednie atrybuty zostaną zmienione w czasie, gdy w kolejce znajdują się komunikaty, kolejność dostarczania jest następująca:

- Kolejność, w jakiej komunikaty są zwracane przez wywołanie MQGET, jest określana na podstawie wartości atrybutów *MsgDeliverySequence* i *DefPriority*, które są wymuszane dla kolejki w czasie, w którym komunikat jest wyświetlany w kolejce:
 - Jeśli parametr *MsgDeliverySequence* ma wartość MQMDS_FIFO po nadejściu komunikatu, komunikat jest umieszczany w kolejce tak, jakby jego priorytetem było *DefPriority*. Nie ma to wpływu na wartość pola *Priority* w deskrytorze komunikatu komunikatu. Pole to zachowuje wartość, jaką miała podczas pierwszego umieszczania komunikatu.
 - Jeśli parametr *MsgDeliverySequence* ma wartość MQMDS_PRIORITY podczas nadejścia komunikatu, komunikat jest umieszczany w kolejce w miejscu właściwym dla priorytetu podanego w polu *Priority* w deskrytorze komunikatu.

Jeśli wartość atrybutu *MsgDeliverySequence* zostanie zmieniona w czasie, gdy w kolejce znajdują się komunikaty, kolejność komunikatów w kolejce nie zostanie zmieniona.

Jeśli wartość atrybutu *DefPriority* zostanie zmieniona w czasie, gdy w kolejce znajdują się komunikaty, komunikaty nie muszą być dostarczane w kolejności FIFO, mimo że atrybut *MsgDeliverySequence* jest ustawiony na wartość MQMDS_FIFO; te, które zostały umieszczone w kolejce z wyższym priorytetem, są dostarczane jako pierwsze.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MSG_DELIVERY_SEQUENCE z wywołaniem MQINQ.

NonPersistentMessageClass (MQLONG)

Cel niezawodności dla nietrwałych komunikatów.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Określa okoliczności, w których nietrwałe komunikaty umieszczone w tej kolejce są odrzucane:

MQNPM_CLASS_NORMAL

Nietrwałe komunikaty są ograniczone do czasu życia sesji menedżera kolejek. Komunikaty te są usuwane w przypadku restartu menedżera kolejek. Ta wartość jest poprawna tylko dla kolejek niewspółużytkowanych i jest to wartość domyślna.

MQNPM_CLASS_HIGH

Menedżer kolejek próbuje zachować nietrwałe komunikaty w czasie życia kolejki. Komunikaty nietrwałe mogą nadal zostać utracone w przypadku niepowodzenia. Ta wartość jest wymuszana dla kolejek współużytkowanych.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NPM_CLASS z wywołaniem MQINQ.

Liczba OpenInput(MQLONG)

Jest to liczba uchwytów, które są obecnie poprawne w przypadku usuwania komunikatów z kolejki za pomocą wywołania MQGET.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Jest to łączna liczba takich uchwytów znanych z *lokalnego* menedżera kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba nie obejmuje otwierania danych wejściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

Liczba ta obejmuje uchwyt, w którym kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla wejścia. Liczba ta nie obejmuje uchwytów, w których kolejka została otwarta dla działań, które nie zawierają danych wejściowych (na przykład kolejka otwarta tylko do przeglądania).

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OPEN_INPUT_COUNT z wywołaniem MQINQ.

Licznik OpenOutput(MQLONG)

Jest to liczba uchwytów, które są obecnie poprawne w przypadku dodawania komunikatów do kolejki za pomocą wywołania MQPUT.

Lokalna	Model	Alias	Zdalny	Klaster
X				

Jest to łączna liczba takich uchwytów znanych dla *lokalnego* menedżera kolejek. Nie obejmuje ona otwierania danych wyjściowych, które zostały wykonane dla tej kolejki w zdalnych menedżerach kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba nie obejmuje operacji otwierania dla danych wyjściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

Liczba ta obejmuje uchwyt, w którym kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla danych wyjściowych. Liczba ta nie obejmuje uchwytów, w których kolejka została otwarta dla działań, które nie zawierają danych wyjściowych (na przykład kolejka otwarta tylko dla zapytania).

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OPEN_OUTPUT_COUNT z wywołaniem MQINQ.

ProcessName (MQCHAR48)

Jest to nazwa obiektu procesu, który jest zdefiniowany w menedżerze kolejek lokalnych. Obiekt procesu identyfikuje program, który może serwisować kolejkę.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Określa sposób obsługi właściwości komunikatu dla komunikatów pobieranych z kolejek przy użyciu wywołania MQGET z opcją MQGMO_PROPERTIES_AS_Q_DEF.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X		

Wartość ta jest jedną z następujących wartości:

MQPROP_ALL

Wszystkie właściwości komunikatu są dołączane wraz z komunikatem, gdy jest on dostarczany do aplikacji. Właściwości te, z wyjątkiem tych, które znajdują się w deskrytorze komunikatu (lub rozszerzeniu), zostają umieszczone w jednym lub większej liczbie nagłówków MQRFH2 danych komunikatu. Jeśli zostanie podany uchwyt komunikatu, zachowanie ma zwrócić właściwości w uchwycie komunikatu.

KOMPATYBILNA_MQPROP_KOMPATYBILNOŚCI

Jeśli wiadomość zawiera właściwość z przedrostkiem mcd., jms., usr. lub mqext., wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2. W przeciwnym razie wszystkie właściwości komunikatu z wyjątkiem tych, które są zawarte w deskrytorze komunikatu lub w rozszerzeniu, są usuwane i nie są już dostępne dla aplikacji. Jest to wartość domyślna. Umożliwia ona aplikacjom, które oczekują, że właściwości związane z produktem JMS znajdują się w nagłówku MQRFH2 w danych komunikatu, aby kontynuować pracę bez modyfikacji. Jeśli zostanie podany uchwyt komunikatu, to zachowanie ma zwrócić właściwości w uchwycie komunikatu.

MQPROP_FORCE_MQRFH2

Właściwości są zawsze zwracane w danych komunikatu w nagłówku MQRFH2 (niezależnie od tego, czy aplikacja określa uchwyt komunikatu). Poprawny uchwyt komunikatu podany w polu MsgHandle w strukturze MQGMO w wywołaniu MQGET jest ignorowany. Właściwości komunikatu nie są dostępne poprzez uchwyt komunikatu.

MQPROP_NONE

Wszystkie właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskrytorze komunikatu (lub rozszerzeniu), są usuwane z komunikatu, zanim komunikat zostanie dostarczony do aplikacji. Jeśli zostanie podany uchwyt komunikatu, zachowanie ma zwrócić właściwości w uchwycie komunikatu.

Ten parametr ma zastosowanie do kolejek lokalnych, aliasowych i modelowych. Aby określić jego wartość, należy użyć selektora MQIA_PROPERTY_CONTROL z wywołaniem MQINQ.

QDepthHighZdarzenie (MQLONG)

Określa, czy generowane są zdarzenia zapętnienia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Zdarzenie Wysokie zapętnienie kolejki wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa progowi wysokiego zapętnienia kolejki (patrz atrybut *QDepthHighLimit*).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Wartość ta jest jedną z następujących wartości:

MQEVN_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVN_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_HIGH_EVENT przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QDepthHighLimit (MQLONG)

Jest to wartość progowa, względem której porównywana jest głębokość kolejki w celu wygenerowania zdarzenia o głębokości kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

To zdarzenie wskazuje, że aplikacja umieściła komunikat w kolejce i że spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wartości progowej zappełnienia kolejki. Patrz [attribut zdarzeniaQDepthHigh](#).

Wartość jest wyrażona jako wartość procentowa maksymalnej głębokości kolejki (attribut *MaxQDepth*) i jest większa lub równa 0 i mniejsza lub równa 100. Wartość domyślna to 80.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_HIGH_LIMIT przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QDepthLowZdarzenie (MQLONG)

Określa, czy generowane są zdarzenia zappełnienia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Zdarzenie Niskie zappełnienie kolejki wskazuje, że aplikacja pobrała komunikat z kolejki i że spowodowało to, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnego progu głębokości kolejki (patrz [QDepthLow](#)).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Wartość ta jest jedną z następujących wartości:

MQEVN_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVN_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_LOW_EVENT przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QDepthLowLimit (MQLONG)

Jest to wartość progowa, względem której porównywana jest głębokość kolejki w celu wygenerowania zdarzenia niedobr kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

To zdarzenie wskazuje, że aplikacja pobrała komunikat z kolejki i że spowodowało to, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnego progu głębokości kolejki. Patrz [atrybut zdarzeniaQDepthLow](#).

Wartość jest wyrażona jako wartość procentowa maksymalnej głębokości kolejki (atrybut *MaxQDepth*) i jest większa lub równa 0 i mniejsza lub równa 100. Wartością domyślną jest 20.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_LOW_LIMIT przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QDepthMaxZdarzenie (MQLONG)

Określa, czy generowane są zdarzenia zapętnienia kolejki. Zdarzenie zapętnienia kolejki wskazuje, że żądanie umieszczenia w kolejce zostało odrzucone, ponieważ kolejka jest pełna, to znaczy, że głębokość kolejki osiągnęła już maksymalną wartość.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Wartość ta jest jedną z następujących wartości:

MQEVR_DISABLED

Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_MAX_EVENT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QDesc (MQCHAR64)

To pole służy do opisowego komentarza.

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	X

Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_DESC przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_DESC_LENGTH.

Nazwa QName (MQCHAR48)

Jest to nazwa kolejki zdefiniowanej w menedżerze kolejek lokalnych.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Wszystkie kolejki zdefiniowane w menedżerze kolejek współużytkuje tę samą przestrzeń nazw kolejki. Oznacza to, że kolejka MQQT_LOCAL i kolejka MQQT_ALIAS nie mogą mieć tej samej nazwy.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Jest to przedział czasu usługi używany do porównania w celu wygenerowania zdarzeń OK dla okresu usługi i okresu usługi OK.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Patrz atrybut zdarzeniaQServiceInterval.

Wartość jest w jednostkach milisekund i jest większa lub równa zero i jest mniejsza lub równa 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_SERVICE_INTERVAL za pomocą wywołania MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale wywołanie MQINQ nie może być używane do określenia jego wartości.

QServiceIntervalZdarzenie (MQLONG)

Określa, czy generowane są zdarzenia OK dla przedziału czasu usługi lub przedziału czasu usługi.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

- Zdarzenie wysokiego interwału usług jest generowane, gdy sprawdzenie wskazuje, że z kolejki nie zostały pobrane żadne komunikaty co najmniej przez czas określony przez atrybut *QServiceInterval*.
- Zdarzenie Interwał usługi OK jest generowane, gdy sprawdzenie wskazuje, że komunikaty zostały pobrane z kolejki w czasie wskazanym przez atrybut *QServiceInterval*.

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Wartość ta jest jedną z następujących wartości:

MQQSIE_WYSOKI

Zdarzenia wysokiego przedziału czasu usługi kolejki są włączone.

- Duże zdarzenia przedziału czasu usługi kolejki są **włączone** i
- Zdarzenia OK przedziału czasu usługi kolejki są **wyłączone**.

MQQSIE_OK

Aktywne zdarzenia przedziału czasu usługi kolejki.

- Duże zdarzenia przedziału czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK interwału usług kolejki są **włączone**.

MQQSIE_NONE

Nie włączono zdarzeń odstępu czasu usługi kolejki.

- Duże zdarzenia przedziału czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK dla przedziału czasu usługi kolejki są także **wyłączone**.

W przypadku kolejek współużytkowanych wartość tego atrybutu jest ignorowana; przyjmuje się wartość MQQSIE_NONE.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_SERVICE_INTERVAL_EVENT z wywołaniem MQINQ.

W systemie z/OS nie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

QSGDisp (MQLONG)

Określa dyspozycję kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	

Wartość ta jest jedną z następujących wartości:

MQQSGD_Q_MGR

Obiekt ma dyspozycję menedżera kolejek. Oznacza to, że definicja obiektu jest znana tylko z lokalnego menedżera kolejek. Definicja ta nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć obiekt o tej samej nazwie i typie, co bieżący obiekt, ale są to oddzielne obiekty i nie istnieje korelacja między nimi. Ich atrybuty nie mogą być takie same, jak w przypadku innych atrybutów.

MQQSGD_COPY

Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć własną kopię tego obiektu. Początkowo wszystkie kopie mają te same atrybuty, ale za pomocą komend MQSC można zmieniać każdą kopię, tak aby jej atrybuty różniły się od tych z pozostałych kopii. Atrybuty kopii są resynchronizowane, gdy definicja wzorca w repozytorium współużytkowanym jest zmieniana.

MQQSGD_SHARED

Obiekt ma współużytkowaną dyspozycję. Oznacza to, że we współużytkowanym repozytorium istnieje pojedyncza instancja obiektu, która jest znana wszystkim menedżerom kolejek w grupie współużytkowania kolejek. Gdy menedżer kolejek w grupie uzyskuje dostęp do obiektu, uzyskuje dostęp do pojedynczej współużytkowanej instancji obiektu.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

QueueAccounting (MQLONG)

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	

Ta opcja steruje kolekcjonowaniem danych rozliczeniowych dla kolejki. W przypadku danych rozliczeniowych, które mają być gromadzone dla tej kolejki, należy również włączyć dane rozliczeniowe dla tego połączenia przy użyciu atrybutu ACCTQ atrybutu QMGR lub pola Opcje w strukturze MQCNO w wywołaniu MQCONN.

Ten atrybut ma jedną z następujących wartości:

MQMON_Q_MGR

Dane rozliczeniowe dla tej kolejki są gromadzone w oparciu o ustawienie atrybutu ACCTQ atrybutu QMGR. Jest to ustawienie domyślne.

MQMON_OFF,

Nie zbieraj danych rozliczeniowych dla tej kolejki.

MQMON_ON

Zbierz dane rozliczeniowe dla tej kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_Q z wywołaniem MQINQ.

QueueMonitoring (MQLONG)

Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Wartość ta jest jedną z następujących wartości:

MQMON_Q_MGR

Zgromaduj dane monitorowania zgodnie z ustawieniem atrybutu menedżera kolejek produktu *QueueMonitoring*. Jest to wartość domyślna.

MQMON_OFF,

Kolekcjonowanie danych monitorowania otwartej bazy danych jest wyłączone dla tej kolejki.

MQMON_LOW

Jeśli wartością atrybutu menedżera kolejek produktu *QueueMonitoring* nie jest MQMON_NONE, włączone jest gromadzenie danych monitorowania w trybie z połączeniem, z niewielką szybkością gromadzenia danych dla tej kolejki.

MQMON_MEDIUM

Jeśli wartością atrybutu menedżera kolejek produktu *QueueMonitoring* jest inny niż MQMON_NONE, włączone jest gromadzenie danych monitorowania w trybie z połączeniem, z umiarkowaną szybkością gromadzenia danych dla tej kolejki.

MQMON_HIGH

Jeśli wartością atrybutu menedżera kolejek produktu *QueueMonitoring* nie jest MQMON_NONE, włączone jest gromadzenie danych monitorowania w trybie z połączeniem, z dużą szybkością gromadzenia danych dla tej kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_Q z wywołaniem MQINQ.

QueueStatistics (MQCHAR12)

Lokalna	Model	Alias	Zdalny	Klaster
X	X	X	X	

Ta opcja steruje gromadzeniem danych statystycznych dla kolejki.

Ten atrybut ma jedną z następujących wartości:

MQMON_Q_MGR

Dane rozliczeniowe dla tej kolejki są gromadzone w oparciu o ustawienie atrybutu STATQ atrybutu QMGR. Jest to ustawienie domyślne.

MQMON_OFF,

Wyłącz gromadzenie danych statystycznych dla tej kolejki.

MQMON_ON

Przełącz gromadzenie danych statystycznych dla tej kolejki.

QType (MQLONG)

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	X

Jest to typ kolejki. Ma jedną z następujących wartości:

MQQT_ALIAS

Definicja kolejki aliasowej.

MQQT_CLUSTER

Kolejka klastra.

MQQT_LOCAL

Kolejka lokalna.

MQQT_REMOTE

Lokalna definicja kolejki zdalnej.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_TYPE z wywołaniem MQINQ.

Nazwa RemoteQMgr(MQCHAR48)

Lokalna	Model	Alias	Zdalny	Klaster
			X	

Jest to nazwa zdalnego menedżera kolejek, w którym zdefiniowana jest kolejka *RemoteQName*. Jeśli kolejka *RemoteQName* ma wartość *QSGDisp* o wartości MQQSGD_COPY lub MQQSGD_SHARED, *RemoteQMgrName* może być nazwą grupy współużytkowania kolejki, która jest właścicielem *RemoteQName*.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, wartość *RemoteQMgrName* nie może być pusta i nie może być nazwą lokalnego menedżera kolejek. Jeśli pole *XmitQName* jest puste, jako kolejka transmisji używana jest kolejka lokalna o takiej samej nazwie, jak nazwa *RemoteQMgrName*. Jeśli nie istnieje kolejka o nazwie *RemoteQMgrName*, używana jest kolejka identyfikowana przez atrybut *DefXmitQName* menedżera kolejek.

Jeśli ta definicja jest używana dla aliasu menedżera kolejek, *RemoteQMgrName* to nazwa menedżera kolejek, który jest aliasem. Może to być nazwa lokalnego menedżera kolejek. W przeciwnym razie, jeśli pole *XmitQName* jest puste w momencie otwarcia, musi istnieć kolejka lokalna o nazwie, która jest taka sama jak *RemoteQMgrName*; Ta kolejka jest używana jako kolejka transmisji.

Jeśli ta definicja jest używana na potrzeby aliasu odpowiedzi, ta nazwa jest nazwą menedżera kolejek, który ma być *ReplyToQMgr*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności dla wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REMOTE_Q_MGR_NAME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Lokalna	Model	Alias	Zdalny	Klaster
			X	

Jest to nazwa kolejki, o której wiadomo, że jest ona znana w zdalnym menedżerze kolejek *RemoteQMgrName*.

Jeśli aplikacja otworzy lokalną definicję kolejki zdalnej, gdy otwarte wystąpi *RemoteQName*, nie może być puste.

Jeśli ta definicja jest używana dla definicji aliasu menedżera kolejek, gdy otwarte występuje *RemoteQName*, musi być puste.

Jeśli definicja jest używana na potrzeby aliasu odpowiedzi, ta nazwa jest nazwą kolejki, która ma być *ReplyToQ*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności dla wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REMOTE_Q_NAME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Jest to okres, w którym ma być zachowana kolejka. Po upływie tego czasu kolejka jest zakwalifikowana do usunięcia.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Czas jest mierzony w godzinach, licząc od daty i godziny utworzenia kolejki. Data i godzina utworzenia kolejki są zapisywane w atrybutach *CreationDate* i *CreationTime*.

Te informacje są udostępniane w celu umożliwienia aplikacji porządkowej lub operatora identyfikowania i usuwania kolejek, które nie są już wymagane.

Uwaga: Menedżer kolejek nigdy nie podejmuje żadnych działań w celu usunięcia kolejek na podstawie tego atrybutu lub w celu uniknięcia usunięcia kolejek z odstępem czasu przechowywania, który nie utracił ważności. Jest to użytkownik odpowiedzialny za podjęcie wszelkich wymaganych działań.

Należy użyć realistycznego przedziału czasu przechowywania, aby zapobiec gromadzeniu trwałych kolejek dynamicznych (patrz *atrybutDefinitionType*). Jednak ten atrybut może być również używany z predefiniowanymi kolejkami.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RETENTION_INTERVAL za pomocą wywołania MQINQ.

Zasięg (MQLONG)

Określa, czy pozycja dla tej kolejki istnieje również w katalogu komórki.

Lokalna	Model	Alias	Zdalny	Klaster
X		X	X	

Katalog komórki jest udostępniany przez usługę nazwy instalowalnej. Wartość ta jest jedną z następujących wartości:

MQSCO_Q_MGR

Definicja kolejki ma zasięg menedżera kolejek: definicja kolejki nie wykracza poza menedżer kolejek, który jest jego właścicielem. Aby otworzyć kolejkę dla danych wyjściowych z innego menedżera kolejek, należy podać nazwę menedżera kolejek będącego właścicielem lub inny menedżer kolejek musi mieć lokalną definicję kolejki.

Komórka MQSCO_CELL

Definicja kolejki ma zasięg komórki: definicja kolejki jest również umieszczana w katalogu komórki, który jest dostępny dla wszystkich menedżerów kolejek w komórce. Kolejka może zostać otwarta dla danych wyjściowych z dowolnego menedżera kolejek w komórce. W tym celu należy określić nazwę kolejki. Nie trzeba podawać nazwy menedżera kolejek, do którego należy kolejka. Definicja kolejki nie jest jednak dostępna dla żadnego menedżera kolejek w komórce, która ma również lokalną definicję kolejki o tej nazwie, ponieważ definicja lokalna ma pierwszeństwo.

Katalog komórki jest udostępniany przez usługę nazwy instalowalnej.

Model i kolejki dynamiczne nie mogą mieć zasięgu komórki.

Ta wartość jest poprawna tylko wtedy, gdy skonfigurowana została usługa nazw obsługując katalog komórek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SCOPE przy użyciu wywołania MQINQ.

Obsługa tego atrybutu podlega następującym ograniczeniom:

- W systemie IBM i atrybut jest obsługiwany, ale poprawna jest tylko wartość MQSCO_Q_MGR.
- W systemie z/OS atrybut nie jest obsługiwany.

Współużytkowalność (MQLONG)

Wskazuje, czy kolejka może być otwierana jednocześnie dla wielu operacji wejściowych.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Wartość ta jest jedną z następujących wartości:

MQQA_SHAREABLE

Kolejka jest współużytkowalna.

Wielokrotne otwarcie z opcją MQOO_INPUT_SHARED jest dozwolone.

MQQA_NOT_SHAREABLE

Kolejka nie jest możliwa do współużytkowania.

Wywołanie MQOPEN z opcją MQOO_INPUT_SHARED jest traktowane jako MQOO_INPUT_EXCLUSIVE.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SHAREABILITY przy użyciu wywołania MQINQ.

StorageClass (MQCHAR8)

Jest to nazwa zdefiniowana przez użytkownika, która definiuje pamięć fizyczną używaną do przechowywania kolejki. W praktyce komunikat jest zapisywany na dysku tylko wtedy, gdy musi być zrzucany z buforu pamięci.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_STORAGE_CLASS z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_STORAGE_CLASS_LENGTH.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

TriggerControl (MQLONG)

Określa, czy komunikaty wyzwalacza są zapisywane w kolejce inicjacji w celu uruchomienia aplikacji w celu obsługi kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Jest to jedna z następujących sytuacji:

MQTC_OFF

Dla tej kolejki nie ma być zapisywane komunikaty wyzwalacza. W tym przypadku wartość *TriggerType* nie ma znaczenia.

MQTC_ON

Komunikaty wyzwalacza mają być zapisywane dla tej kolejki po wystąpieniu odpowiednich zdarzeń wyzwalających.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_CONTROL przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerData (MQCHAR64)

Jest to dane w formacie wolnym, które menedżer kolejek wstawia do komunikatu wyzwalacza, gdy komunikat przybywający do tej kolejki powoduje zapisanie komunikatu wyzwalacza w kolejce inicjuj.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Treść tych danych nie ma znaczenia dla menedżera kolejek. Ma znaczenie dla aplikacji monitorującego wyzwalacz, która przetwarza kolejkę inicjującą, lub do aplikacji, która jest uruchamiana przez monitor wyzwalacza.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_TRIGGER_DATA przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET. Długość tego atrybutu jest podana przez wartość MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Jest to liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej, które muszą znajdować się w kolejce, zanim zostanie zapisany komunikat wyzwalacza. Ma to zastosowanie, gdy parametr *TriggerType* jest ustawiony na wartość MQTT_DEPTH. Wartość *TriggerDepth* jest równa lub większa od jednej. Ten atrybut nie jest używany w inny sposób.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_DEPTH przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Priorytet TriggerMsg(MQLONG)

Jest to priorytet komunikatu, poniżej którego komunikaty nie przyczyniają się do generowania komunikatów wyzwalacza (oznacza to, że menedżer kolejek ignoruje te komunikaty przy podejmowaniu decyzji o wygenerowaniu komunikatu wyzwalacza).

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

TriggerMsgPriority może być w zakresie od zera (od najniższego) do *MaxPriority* (najwyższy; patrz [atributMaxPriority](#)); wartość zero powoduje, że wszystkie komunikaty mogą przyczyniać się do generowania komunikatów wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_MSG_PRIORITY przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerType (MQLONG)

Określa to warunki, w których komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Ma jedną z następujących wartości:

MQTT_NONE

Żadne komunikaty wyzwalacza nie są zapisywane w wyniku komunikatów w tej kolejce. Ma to ten sam efekt, co ustawienie *TriggerControl* na wartość MQTT_OFF.

MQTT_FIRST

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce zmienia się z zakresu od 0 do 1.

MQTT EVERY

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy w kolejce pojawia się komunikat o priorytecie *TriggerMsgPriority* lub wyższym.

MQTT_DEPTH

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce jest równa lub większa niż *TriggerDepth*. Po zapisaniu komunikatu wyzwalacza opcja *TriggerControl* jest ustawiona na wartość MQTT_OFF, aby zapobiec kolejnym wyzwalaniu, dopóki nie zostanie ona jawnie włączona.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_TYPE z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Użycie (MQLONG)

Określa, dla której kolejki jest używana.

Lokalna	Model	Alias	Zdalny	Klaster
X	X			

Wartość ta jest jedną z następujących wartości:

MQUS_NORMAL

Jest to kolejka, której aplikacje używają podczas umieszczania i pobierania komunikatów. Kolejka nie jest kolejką transmisji.

MQUS_TRANSMISSION

Jest to kolejka używana do przechowywania komunikatów przeznaczonych dla menedżerów kolejek zdalnych. Gdy aplikacja wysyła komunikat do kolejki zdalnej, lokalny menedżer kolejek przechowuje komunikat tymczasowo w odpowiedniej kolejce transmisji w specjalnym formacie. Agent kanału komunikatów odczytuje następnie komunikat z kolejki transmisji i transportuje komunikat do zdalnego menedżera kolejek. Więcej informacji na temat kolejek transmisji zawiera sekcja [Definiowanie kolejki transmisji](#).

Tylko aplikacje uprzywilejowane mogą otwierać kolejkę transmisji dla komendy MQOO_OUTPUT w celu bezpośredniego umieszczania komunikatów na niej. Zwykle są to tylko aplikacje narzędziowe. Upewnij się, że format danych komunikatu jest poprawny (patrz „MQXQH-nagłówek kolejki transmisji” na stronie 623) lub błędy mogą wystąpić w trakcie procesu transmisji. Kontekst nie jest przekazywany ani ustawiany, chyba że zostanie podana jedna z opcji kontekstu MQPMO_*_CONTEXT.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_USAGE z wywołaniem MQINQ.

XmitQName (MQCHAR48)

Jest to nazwa kolejki transmisji. Jeśli ten atrybut jest niepusty w przypadku otwarcia, dla kolejki zdalnej lub definicji aliasu menedżera kolejek, określa ona nazwę lokalnej kolejki transmisji, która ma być używana do przekazywania komunikatu.

Lokalna	Model	Alias	Zdalny	Klaster
			X	

Jeśli pole *XmitQName* jest puste, jako kolejka transmisji używana jest kolejka lokalna o nazwie, która jest taka sama, jak nazwa *RemoteQMGrName*. Jeśli nie istnieje kolejka o nazwie *RemoteQMGrName*, używana jest kolejka identyfikowana przez atrybut *DefXmitQName* menedżera kolejek.

Ten atrybut jest ignorowany, jeśli definicja jest używana jako alias menedżera kolejek, a *RemoteQMgrName* to nazwa lokalnego menedżera kolejek. Atrybut nie jest również brany pod uwagę, jeśli definicja jest używana jako definicja aliasu kolejki zwrotnej.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_XMIT_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_Q_NAME_LENGTH.

Atrybuty dla list nazw

W poniższej tabeli przedstawiono podsumowanie atrybutów, które są specyficzne dla list nazw. Atrybuty są opisane w kolejności alfabetycznej.

Listy nazw są obsługiwane we wszystkich systemach IBM MQ oraz IBM MQ MQI clients połączonych z tymi systemami.

Uwaga: Nazwy atrybutów wyświetlane w tej sekcji to nazwy opisowe używane w wywołaniach MQINQ i MQSET; nazwy te są takie same, jak w przypadku komend PCF. Jeśli komendy MQSC są używane do definiowania, modyfikowania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja [Komendy skryptowe \(MQSC\)](#).

Tabela 115. Atrybuty dla list nazw	
Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
NameCount	Liczba nazw na liście nazw
NamelistDesc	Opis listy nazw
NamelistName	Nazwa listy nazw
Nazwy	Lista nazw <i>NameCount</i>
NamelistType	Typ listy nazw
QSGDISP	Dyspozycja grupy współużytkowania kolejki

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_TIME_LENGTH.

NameCount (MQLONG)

Liczba nazw na liście nazw. Wartość ta jest większa lub równa zero. Zdefiniowana jest następująca wartość:

Nr_NAME_NAME_COUNT MQNC_MAX_NAME_
Maksymalna liczba nazw na liście nazw.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NAME_COUNT z wywołaniem MQINQ.

NamelistDesc (MQCHAR64)

To pole służy do opisowego komentarza; jego wartość jest ustanawiana przez proces definiowania. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero;

jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole to może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMELIST_DESC przy użyciu wywołania MQINQ.

Długość tego atrybutu jest podana przez wartość MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Jest to nazwa listy nazw, która jest zdefiniowana w menedżerze kolejek lokalnych. Więcej informacji na temat nazw list nazw znajduje się w sekcji Nazwy innych obiektów.

Każda lista nazw ma inną nazwę niż nazwy innych list nazw należących do menedżera kolejek, ale może duplikować nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMELIST_NAME w wywołaniu MQINQ.

Długość tego atrybutu jest podana przez wartość MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Określa rodzaj nazw na liście nazw i wskazuje, w jaki sposób używana jest lista nazw. Jest to jedna z następujących wartości:

MQNT_NONE

Lista nazw bez przypisanego typu.

MQNT_Q

Lista nazw zawierająca nazwy kolejek.

MQNT_CLUSTER

Lista nazw zawierająca nazwy klastrów.

MQNT_AUTH_INFO

Lista nazw zawierająca nazwy obiektów informacji uwierzytelniających.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NAMELIST_TYPE z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Nazwy (MQCHAR48xNameCount)

Jest to lista nazw *NameCount*, gdzie każda nazwa jest nazwą obiektu, który jest zdefiniowany w lokalnym menedżerze kolejek. Więcej informacji na temat nazw obiektów zawiera sekcja Reguły nazewnictwa obiektów IBM MQ.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMES przy użyciu wywołania MQINQ.

Długość każdej nazwy na liście jest podana przez wartość MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Ta opcja określa dyspozycję listy nazw. Wartość ta jest jedną z następujących wartości:

MQQSGD_Q_MGR

Obiekt ma dyspozycję menedżera kolejek: definicja obiektu jest znana tylko z lokalnego menedżera kolejek. Definicja ta nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć obiekt o tej samej nazwie i typie, co bieżący obiekt, ale są to oddzielne obiekty i nie istnieje korelacja między nimi. Ich atrybuty nie mogą być takie same, jak w przypadku innych atrybutów.

MQQSGD_COPY

Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć własną kopię tego obiektu. Początkowo wszystkie kopie mają te same atrybuty, ale każda kopia może być zmieniona za pomocą komend MQSC, tak aby jego atrybuty różniły się od tych z pozostałych kopii. Atrybuty kopii są resynchronizowane, gdy definicja wzorca w repozytorium współużytkowanym jest zmieniana.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

Atrybuty definicji procesów

W poniższej tabeli podsumowano atrybuty specyficzne dla definicji procesów. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy atrybutów w tej sekcji to nazwy opisowe używane w wywołaniach MQINQ i MQSET; nazwy te są takie same, jak w przypadku komend PCF. Jeśli komendy MQSC są używane do definiowania, modyfikowania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja [Komendy skryptowe \(MQSC\)](#).

Tabela 116. Atrybuty definicji procesów	
Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
ApplId	Identyfikator aplikacji
ApplType	Typ aplikacji
EnvData	Dane środowiska
ProcessDesc	Opis procesu
ProcessName	Nazwa procesu
QSGDISP	Dyspozycja grupy współużytkowania kolejki
UserData	Dane użytkownika

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. Te informacje są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalacza.

Znaczenie *ApplId* jest określone przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez produkt IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego. Następujące uwagi mają zastosowanie do wskazanych środowisk:

- W systemie z/OS, *ApplId* musi być:

- Identyfikator transakcji CICS , dla aplikacji uruchomionych za pomocą wyzwalacza CICS
-monitorowanie transakcji CKTI
- Identyfikator transakcji IMS dla aplikacji uruchamianych przy użyciu monitora wyzwalacza IMS
CSQQTRMN
- W systemach Windows nazwa programu może być poprzedzona ścieżką napędu i ścieżką do katalogu.
- W systemie Systemy UNIXnazwa programu może być poprzedzona ścieżką do katalogu.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_APPL_ID z wywołaniem MQINQ. Długość tego atrybutu jest podana przez MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Identyfikuje rodzaj programu, który ma być uruchomiony w odpowiedzi na odezwie komunikatu wyzwalacza. Te informacje są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalacza.

ApplType może mieć dowolną wartość, ale dla typów standardowych zalecane są następujące wartości; ogranicz typy aplikacji zdefiniowane przez użytkownika do wartości z zakresu MQAT_USER_FIRST za pomocą MQAT_USER_LAST:

MQAT_AIX

Aplikacja AIX (ta sama wartość jak MQAT_UNIX).

MQAT_BATCH

aplikacja wsadowa

MQAT_BROKER

Aplikacja brokera

MQAT_CICS

CICS .

MQAT_CICS_BRIDGE

Aplikacja CICS bridge .

MQAT_CICS_VSE

CICS/VSE .

MQAT_DOS

Aplikacja IBM MQ MQI client na komputerze PC DOS.

MQAT_IMS

Aplikacja IMS .

MQAT_IMS_BRIDGE

Aplikacja pomostowa IMS .

MQAT_JAVA

Aplikacja Java .

MQAT_MVS

MVS lub aplikacji TSO (taka sama wartość jak MQAT_ZOS).

MQAT_NOTES_AGENT

Lotus Notes Aplikacja agenta.

MQAT_NSK

Aplikacja HP Integrity NonStop Server .

MQAT_OS390

Aplikacja OS/390 (taka sama wartość jak MQAT_ZOS).

MQAT_OS400

Aplikacja IBM i .

MQAT_RRS_BATCH

Aplikacja wsadowa RRS.

MQAT_UNIX

Aplikacja UNIX .

MQAT_UNKNOWN

Aplikacja o nieznanym typie.

UŻYTKOWNIKA_MQAT_

Aplikacja użytkownika.

MQAT_VOS

Aplikacja Stratus VOS.

MQAT_WINDOWS

16-bitowa aplikacja Windows .

MQAT_WINDOWS_NT

32-bitowa aplikacja Windows .

MQAT_WLM

Aplikacja menedżera obciążenia produktu z/OS .

MQAT_XCF

XCF.

MQAT_ZOS

Aplikacja z/OS .

MQAT_USER_FIRST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

MQAT_USER_LAST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_APPL_TYPE z wywołaniem MQINQ.

EnvData (MQCHAR128)

Jest to łańcuch znaków zawierający informacje dotyczące środowiska dotyczące aplikacji, która ma zostać uruchomiona. Te informacje są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalacza.

Znaczenie *EnvData* jest określone przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez program IBM MQ dopisuje *EnvData* do listy parametrów przekazanej do uruchomionej aplikacji. Lista parametrów składa się ze struktury MQTMC2 , po której następują jedno puste, po którym następuje *EnvData* z usuniętym odstępami końcowymi. Następujące uwagi mają zastosowanie do wskazanych środowisk:

- w t.z/OS:
 - Produkt *EnvData* nie jest używany przez aplikacje monitora wyzwalacza udostępniane przez produkt IBM MQ.
 - Jeśli parametr ApplType ma wartość MQAT_WLM, można podać wartości domyślne w polach EnvData dla pól ServiceName i ServiceStep w nagłówku informacji o pracy (MQWIH).
- W systemie Systemy UNIX można ustawić *EnvData* na znak & , aby uruchomić uruchomioną aplikację w tle.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ENV_DATA przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez wartość MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

To pole służy do opisowego komentarza. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierano tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_DESC przy użyciu wywołania MQINQ.

Długość tego atrybutu jest podana przez wartość MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Jest to nazwa definicji procesu, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda definicja procesu ma nazwę różniącą się od nazw innych definicji procesów należących do menedżera kolejek. Jednak nazwa definicji procesu może być taka sama, jak nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_NAME z wywołaniem MQINQ.

Długość tego atrybutu jest podana przez MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Określa dyspozycję definicji procesu. Wartość ta jest jedną z następujących wartości:

MQQSGD_Q_MGR

Obiekt ma dyspozycję menedżera kolejek: definicja obiektu jest znana tylko z lokalnego menedżera kolejek. Definicja ta nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć obiekt o tej samej nazwie i typie, co bieżący obiekt, ale są to oddzielne obiekty i nie istnieje korelacja między nimi. Ich atrybuty nie mogą być takie same, jak w przypadku innych atrybutów.

MQQSGD_COPY

Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć własną kopię tego obiektu. Początkowo wszystkie kopie mają te same atrybuty, ale każda kopia może być zmieniona za pomocą komend MQSC, tak aby jego atrybuty różniły się od tych z pozostałymi kopiami. Atrybuty kopii są resynchronizowane, gdy definicja wzorca w repozytorium współużytkowanym jest zmieniana.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

UserData (MQCHAR128)

UserData jest łańcuchem znaków, który zawiera informacje o użytkowniku dotyczące aplikacji, która ma zostać uruchomiona. Informacje te są przeznaczone do użycia przez aplikację monitorującego wyzwania, która przetwarza komunikaty w kolejce inicjuj lub aplikacji uruchomionej przez monitor wyzwacza. Informacje te są wysyłane do kolejki inicjuj jako część komunikatu wyzwacza.

Znaczenie *UserData* jest określane przez aplikację wyzwacza-monitor. Monitor wyzwacza udostępniony przez produkt IBM MQ przekazuje produkt *UserData* do uruchomionej aplikacji jako część listy parametrów. Lista parametrów składa się ze struktury MQTMC2 (zawierającej *UserData*), po której następuje jedno puste miejsce, po którym następuje *EnvData* z usuniętą spacjami kończącymi.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi. W przypadku systemu Microsoft Windows łańcuch znaków nie może zawierać podwójnych cudzysłowów, jeśli definicja procesu ma być przekazana do produktu **runmqtrm**.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_USER_DATA z wywołaniem MQINQ. Długość tego atrybutu jest podana przez wartość MQ_PROCESS_USER_DATA_LENGTH.

Kody powrotu

Dla każdego wywołania interfejsu IBM MQ (MQI) i interfejsu administracyjnego IBM MQ (MQAI): kod **ukończenia** i kod **przyczyna** są zwracane przez menedżer kolejek lub przez procedurę wyjścia, w celu wskazania powodzenia lub niepowodzenia wywołania.

Aplikacje nie mogą zależeć od błędów, które są sprawdzane w określonej kolejności, z wyjątkiem przypadków, w których zaznaczono inaczej. Jeśli z wywołania może powstać więcej niż jeden kod zakończenia lub kod przyczyny, konkretny zgłoszony błąd zależy od implementacji.

Aplikacje sprawdzające pomyślne zakończenie po wywołaniu funkcji API IBM MQ muszą zawsze sprawdzać kod zakończenia. Nie należy zakładać wartości kodu zakończenia w oparciu o wartość kodu przyczyny.

Kody zakończenia

Parametr kodu zakończenia (*CompCode*) pozwala programowi wywołującemu na szybkie sprawdzenie, czy wywołanie zostało zakończone pomyślnie, zakończone częściowo lub nie powiodło się. Poniżej znajduje się lista kodów zakończenia, z bardziej szczegółowym opisem, niż podano w opisach wywołań:

MQCC_OK

Wywołanie zakończyło się całkowicie; wszystkie parametry wyjściowe zostały ustawione. W tym przypadku parametr *Reason* zawsze ma wartość MQRC_NONE.

MQCC_WARNING,

Połączenie zostało zakończone częściowo. Niektóre parametry wyjściowe mogły zostać ustawione jako uzupełnienie parametrów wyjściowych *CompCode* i *Reason*. Parametr *Reason* zawiera dodatkowe informacje o częściowym zakończeniu.

MQCC_FAILED

Przetwarzanie wywołania nie zostało zakończone. Stan menedżera kolejek pozostaje niezmieniony, z wyjątkiem sytuacji, w których zaznaczono inaczej. Parametry wyjściowe *CompCode* i *Reason* zostały ustawione; pozostałe parametry są niezmienione, z wyjątkiem sytuacji, w których zaznaczono.

Przyczyną może być błąd w programie użytkowym lub jego wynik może być wynikiem sytuacji zewnętrznej dla programu, na przykład uprawnienia użytkownika mogły zostać odwołane. Parametr *Reason* zawiera dodatkowe informacje na temat błędu.

Kody przyczyny

Parametr kodu przyczyny (*Reason*) kwalifikuje parametr kodu zakończenia (*CompCode*).

Jeśli nie ma specjalnego powodu do raportowania, zwracana jest wartość MQRC_NONE. Pomyślne wywołanie zwraca MQCC_OK i MQRC_NONE.

Jeśli kodem zakończenia jest MQCC_WARNING lub MQCC_FAILED, menedżer kolejek zawsze zgłasza odpowiedni powód; szczegóły są podawane w każdym opisie wywołania.

W przypadku, gdy procedury obsługi wyjścia użytkownika ustawiają kody zakończenia i przyczyny, muszą być zgodne z tymi regułami. Ponadto wszystkie specjalne wartości przyczyny zdefiniowane przez procedury zewnętrzne muszą być mniejsze od zera, aby zapewnić, że nie będą one kolidowały z wartościami zdefiniowanymi przez menedżer kolejek. Wyjścia mogą ustawiać przyczyny już zdefiniowane przez menedżer kolejek, jeśli jest to konieczne.

Kody przyczyny występują również w:

- Pole *Reason* struktury MQDLH

- Pole *Feedback* struktury MQMD

Pełne opisy kodów przyczyny znajdują się w sekcji [Komunikaty i kody przyczyny](#).

Reguły sprawdzania poprawności opcji MQI

Ta sekcja zawiera listę sytuacji, które generują kod przyczyny MQRC_OPTIONS_ERROR z wywołania MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE lub MQSUB.

Wywołanie MQOPEN

Dla opcji wywołania MQOPEN:

- Należy określić co najmniej *jeden* z następujących elementów:

- MQOO_BROWSE
- MQOO_INPUT_EXCLUSIVE ¹
- MQOO_INPUT_SHARED ¹
- MQOO_INPUT_AS_Q_DEF ¹
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_ON_OPEN ²
- MQOO_BIND_NOT_FIXED ²
- MQOO_BIND_ON_GROUP ²
- MQOO_BIND_AS_Q_DEF ²

- Dozwolony jest tylko *jeden* z następujących:

- MQOO_READ_AHEAD
- MQOO_NO_READ_AHEAD
- MQOO_READ_AHEAD_AHEAD_AS_Q_DEF

1. Dozwolony jest tylko *jeden* z następujących:

- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_SHARED
- MQOO_INPUT_AS_Q_DEF

2. Dozwolony jest tylko *jeden* z następujących:

- MQOO_BIND_ON_OPEN
- MQOO_BIND_NOT_FIXED
- MQOO_BIND_ON_GROUP
- MQOO_BIND_AS_Q_DEF

Uwaga: Opcje, które zostały wymienione wcześniej, wzajemnie się wykluczają. Jednak ponieważ wartość parametru MQOO_BIND_AS_Q_DEF jest równa zero, określenie tej wartości przy użyciu jednej z dwóch pozostałych opcji wiązania nie powoduje wystąpienia kodu przyczyny MQRC_OPTIONS_ERROR. Komenda MQOO_BIND_AS_Q_DEF jest udostępniana w celu uzyskania dokumentacji programu pomocowego.

- Jeśli określono parametr MQOO_SAVE_ALL_CONTEXT, należy również określić jedną z opcji MQOO_INPUT_ *.
- Jeśli określono jedną z opcji MQOO_SET_ * _CONTEXT lub MQOO_PASS_ * _CONTEXT, należy również określić parametr MQOO_OUTPUT.
- Jeśli określono parametr MQOO_CO_OP, należy również określić parametr MQOO_BROWSE.

- Jeśli określono parametr MQOO_NO_MULTICAST, należy również określić parametr MQOO_OUTPUT.

Wywołanie MQPUT

Dla opcji put-message:

- Kombinacja MQPMO_SYNCPOINT i MQPMO_NO_SYNCPOINT nie jest dozwolona.
- Dozwolony jest tylko *jeden* z następujących:
 - MQPMO_DEFAULT_CONTEXT
 - MQPMO_NO_CONTEXT
 - MQPMO_PASS_ALL_CONTEXT
 - MQPMO_PASS_IDENTITY_CONTEXT
 - MQPMO_SET_ALL_CONTEXT
 - MQPMO_SET_IDENTITY_CONTEXT
- Dozwolony jest tylko *jeden* z następujących:
 - MQPMO_ASYNC_RESPONSE
 - MQPMO_SYNC_RESPONSE
 - MQPMO_RESPONSE_AS_TOPIC_DEF
 - MQPMO_RESPONSE_AS_Q_DEF
- Parametr MQPMO_ALTERNATE_USER_AUTHORITY jest niedozwolony (jest on poprawny tylko w wywołaniu MQPUT1).

Wywołanie MQPUT1

W przypadku opcji put-message reguły są takie same, jak dla wywołania MQPUT, z wyjątkiem następujących:

- Uprawnienie MQPMO_ALTERNATE_USER_AUTHORITY jest dozwolone.
- Parametr MQPMO_LOGICAL_ORDER *nie* jest dozwolony.

Wywołanie MQGET

Dla opcji get-message:

- Dozwolony jest tylko *jeden* z następujących:
 - MQGMO_NO_SYNCPOINT
 - MQGMO_SYNCPOINT,
 - MQGMO_SYNCPOINT_IF_PERSISTENT
- Dozwolony jest tylko *jeden* z następujących:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_MSG_UNDER_CURSOR
- Parametr MQGMO_SYNCPOINT nie jest dozwolony dla następujących elementów:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - Blokada MQGMO_LOCK
 - MQGMO_UNLOCK

- Parametr MQGMO_SYNCPOINT_IF_PERSISTENT jest niedozwolony z następującymi:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
 - MQGMO_COMPLETE_MSG
 - MQGMO_UNLOCK
- Parametr MQGMO_MARK_SKIP_BACKOUT wymaga określenia MQGMO_SYNCPOINT.
- Kombinacja MQGMO_WAIT i MQGMO_SET_SIGNAL nie jest dozwolona.
- Jeśli określono parametr MQGMO_LOCK, należy również określić jedną z następujących wartości:
 - MQGMO_BROWSE_FIRST
 - MQGMO_BROWSE_MSG_UNDER_CURSOR
 - MQGMO_BROWSE_NEXT
- Jeśli określono parametr MQGMO_UNLOCK, dozwolone są tylko następujące wartości:
 - MQGMO_NO_SYNCPOINT
 - MQGMO_NO_WAIT

Wywołanie MQCLOSE

Aby uzyskać informacje na temat opcji wywołania MQCLOSE:

- Kombinacja MQCO_DELETE i MQCO_DELETE_PURGE nie jest dozwolona.
- Dozwolona jest tylko jedna z następujących wartości:
 - MQCO_KEEP_SUB
 - MQCO_REMOVE_SUB

Wywołanie MQSUB

W przypadku opcji wywołania MQSUB:

- Należy określić co najmniej jedną z następujących wartości:
 - MQSO_ALTER
 - MQSO_RESUME
 - MQSO_CREATE
- Dozwolona jest tylko jedna z następujących wartości:
 - MQSO_DURABLE,
 - MQSO_NON_DURABLE,

Uwaga: Opcje, które zostały wymienione wcześniej, wzajemnie się wykluczają. Jednak ponieważ wartość parametru MQSO_NON_DURABLE jest równa zero, określenie jej przy użyciu parametru MQSO_DURABLE nie powoduje wystąpienia kodu przyczyny MQRC_OPTIONS_ERROR. Parametr MQSO_NON_DURABLE jest udostępniany w celu uzyskania dokumentacji programu pomocy.

- Kombinacja opcji MQSO_GROUP_SUB i MQSO_MANAGED nie jest dozwolona.
- Parametr MQSO_GROUP_SUB wymaga określenia wartości MQSO_SET_CORREL_ID.
- Dozwolona jest tylko jedna z następujących wartości:
 - MQSO_ANY_USERID
 - MQSO_FIXED_USERID
- MQSO_NEW_PUBLICATIONS_ONLY jest dozwolone w połączeniu z:
 - MQSO_CREATE

- MQSO_ALTER, jeśli ustawiono parametr MQSO_NEW_PUBLICATIONS_ONLY na oryginalnej subskrypcji
- Kombinacja parametrów MQSO_PUBLICATIONS_ON_REQUEST i SubLevel większa niż 1 nie jest dozwolona.
- Dozwolona jest tylko jedna z następujących wartości:
 - MQSO_WILDCARD_CHAR
 - MQSO_WILDCARD_TOPIC
- Parametr MQSO_NO_MULTICAST wymaga podania parametru MQSO_MANAGED.

Komunikaty w kolejce publikowania/subskrypcji w kolejce

Aplikacja może używać komunikatów komend produktu MQRFH2 do sterowania kolejką aplikacji publikowania/subskrypcji.

Aplikacja, która używa produktu MQRFH2 do publikowania/subskrypcji, może wysyłać następujące komunikaty komend do systemu SYSTEM.BROKER.CONTROL.QUEUE:

- [“Usuń komunikat o publikacji” na stronie 889](#)
- [“Komunikat subskrybenta programu Deregister” na stronie 890](#)
- [“Publikuj komunikat” na stronie 895](#)
- [“Rejestrowanie komunikatu subskrybenta” na stronie 897](#)
- [“Komunikat o aktualizacji żądania” na stronie 902](#)

W przypadku zapisu w kolejce aplikacji publikowania/subskrypcji należy zapoznać się z tymi komunikatami, komunikatem odpowiedzi menedżera kolejek oraz deskryptorem komunikatu (MQMD). Informacje na ten temat zawierają następujące informacje:

- [“Komunikat odpowiedzi menedżera kolejek” na stronie 904](#)
- [“Ustawienia MQMD dla publikacji przekazywanych przez menedżera kolejek” na stronie 910](#)
- [“Ustawienia MQMD w komunikatach odpowiedzi menedżera kolejek” na stronie 911](#)
- [“Kody przyczyny publikowania/subskrypcji” na stronie 906](#)

Komendy te znajdują się w folderze <psc> w polu **NameValueData** nagłówka MQRFH2. Komunikat, który może zostać wysłany przez brokera w odpowiedzi na komunikat komendy, znajduje się w folderze <pscrt>.

Opisy poszczególnych komend zawierają listę właściwości, które mogą być zawarte w folderze. Jeśli nie określono inaczej, właściwości są opcjonalne i mogą wystąpić tylko raz.

Nazwy właściwości są wyświetlane jako <Command>.

Wartości muszą być w formacie łańcucha, na przykład: Publish.

Stała łańcuchowa reprezentująca wartość właściwości jest wyświetlana w nawiasach, na przykład: (MQPSC_PUBLISH).

Stałe łańcuchowe są zdefiniowane w pliku nagłówkowego cmqpsc.h, który jest dostarczany z menedżerem kolejek.

Usuń komunikat o publikacji

Komunikat komendy **Delete Publication** jest wysyłany do menedżera kolejek z publikatora lub z innego menedżera kolejek w celu poinformowania menedżera kolejek o usunięciu wszelkich zachowanych publikacji dla określonych tematów.

Ten komunikat jest wysyłany do kolejki monitorowanej przez interfejs kolejki publikowania/subskrybowania w kolejce menedżera kolejek.

Kolejka wejściowa powinna być kolejką, do której została wysłana oryginalna publikacja.

Jeśli użytkownik ma uprawnienia do niektórych, ale nie wszystkich tematów, które są określone w komunikacie komendy **Delete Publication**, to tylko te tematy są usuwane. Komunikat **Broker Response** wskazuje, które tematy nie są usuwane.

Podobnie, jeśli komenda **Publish** zawiera więcej niż jeden temat, komenda **Delete Publication**, która jest zgodna z niektórymi, ale nie wszystkimi, powoduje usunięcie tylko tych publikacji dla tematów, które zostały określone w komendzie **Delete Publication**.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD), które są wymagane podczas wysyłania komunikatu komendy do menedżera kolejek, zawiera sekcja [“Ustawienia MQMD dla publikacji przekazywanych przez menedżera kolejek”](#) na stronie 910.

Właściwości

< komenda> (MQPSC_COMMAND)

Ta wartość to DeletePub (MQPSC_DELETE_PUBLICATION).

Ta właściwość musi być określona.

< Topic> (MQPSC_TOPIC)

Wartość jest łańcuchem zawierającym temat, dla którego przechowywane publikacje mają zostać usunięte. Znaki wieloznaczne mogą być zawarte w łańcuchu w celu usunięcia publikacji dotyczących więcej niż jednego tematu.

Ta właściwość musi być określona. Można ją powtarzać w razie potrzeby w przypadku wielu tematów.

<DelOpt> (MQPSC_DELETE_OPTION)

Właściwość opcji usuwania może przyjmować jedną z następujących wartości:

Lokalna (MQPSC_LOCAL)

Wszystkie zachowane publikacje dotyczące określonych tematów są usuwane z lokalnego menedżera kolejek (czyli menedżera kolejek, do którego wysyłany jest ten komunikat), bez względu na to, czy zostały one opublikowane z opcją Lokalnie, czy nie.

Nie ma to wpływu na publikacje w innych menedżerach kolejek.

Brak (MQPSC_NONE)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości DelOpt. Jeśli w tym samym czasie zostaną podane inne opcje, opcja Brak zostanie zignorowana.

Wartością domyślną, jeśli ta właściwość jest pominięta, jest to, że wszystkie zachowane publikacje dla określonych tematów są usuwane we wszystkich menedżerach kolejek w sieci, niezależnie od tego, czy zostały one opublikowane za pomocą opcji Lokalnie.

Przykład

Poniżej znajduje się przykład danych NameValueData dla komunikatu komendy **Delete Publication**. Jest ona używana przez przykładową aplikację do usunięcia w lokalnym menedżerze kolejek, zachowanej publikacji, która zawiera najnowszy wynik w zgodności między Team1 i Team2.

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

Komunikat subskrybenta programu Deregister

Komunikat komendy **Deregister Subscriber** jest wysyłany do menedżera kolejek przez subskrybent lub przez inną aplikację w imieniu subskrybenta, aby wskazać, że nie chce już odbierać komunikatów zgodnych z podanymi parametrami.

Ten komunikat jest wysyłany do systemu SYSTEM.BROKER.CONTROL.QUEUE, kolejka sterująca menedżera kolejek. Użytkownik musi mieć uprawnienia niezbędne do umieszczenia komunikatu w tej kolejce.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD), które są wymagane podczas wysyłania komunikatu komendy do menedżera kolejek, zawiera sekcja Ustawienia MQMD dla publikacji przekazywanych przez menedżera kolejek.

Poszczególne subskrypcje można wyrejestrować, określając odpowiedni temat, punkt subskrypcji i wartości filtru dla pierwotnej subskrypcji. Jeśli którekolwiek z wartości nie zostały określone (to znaczy, że przyjmowały wartości domyślne) w pierwotnej subskrypcji, należy je pominąć, gdy subskrypcja jest wyrejestrowywana.

Wszystkie subskrypcje subskrybenta lub grupy subskrybentów mogą zostać wyrejestrowywane za pomocą opcji `DeregAll`. Jeśli na przykład zostanie określony parametr `DeregAll`, razem z punktem subskrypcji (ale bez tematu lub filtru), wszystkie subskrypcje subskrybenta w określonym punkcie subskrypcji zostaną wyrejestrowywane, niezależnie od tematu i filtru. Dozwolona jest dowolna kombinacja tematu, filtru i punktu subskrypcji. Jeśli wszystkie trzy są określone tylko w jednej subskrypcji, to opcja `DeregAll` jest ignorowana.

Komunikat musi zostać wysłany przez subskrybenta, który zarejestrował subskrypcję. Ten komunikat jest potwierdzany przez sprawdzenie identyfikatora użytkownika subskrybenta.

Za pomocą komend `MQSC` lub `PCF` subskrypcje mogą być również wyrejestrowywać przez administratora systemu. Jednak subskrypcje zarejestrowane w tymczasowej kolejce dynamicznej są powiązane z kolejką, a nie tylko nazwą kolejki. Jeśli kolejka została usunięta, jawnie lub przez aplikację odłączającą się od menedżera kolejek, nie jest już możliwe użycie komendy **Deregister Subscriber** w celu wyrejestrowania subskrypcji dla tej kolejki. Subskrypcje można wyrejestrować za pomocą środowiska roboczego dla programisty. Są one usuwane automatycznie przez menedżer kolejek przy następnym dopasowaniu publikacji do subskrypcji lub po następnym restarcie menedżera kolejek. W normalnych okolicznościach aplikacje powinny wyrejestrować swoje subskrypcje przed usunięciem kolejki lub odłączając się od menedżera kolejek.

Jeśli subskrybent wysła komunikat w celu wyrejestrowania subskrypcji i otrzyma komunikat odpowiedzi, aby stwierdzić, że został on pomyślnie przetworzony, niektóre publikacje mogą nadal dotrzeć do kolejki subskrybenta, jeśli były przetwarzane przez menedżer kolejek w tym samym czasie, w którym subskrypcja została wyrejestrowana. Jeśli komunikaty nie zostaną usunięte z kolejki, może to być kompilacja nieprzetworzonych komunikatów w kolejce subskrybenta. Jeśli aplikacja wykonuje pętlę, która zawiera wywołanie `MQGET` z odpowiednim `CorrelId` po spaniu przez pewien czas, te komunikaty są usuwane z kolejki.

Podobnie, jeśli subskrybent używa trwałej kolejki dynamicznej i wyrejestrowywania i zamyka kolejkę za pomocą opcji `MQCO_DELETE_PURGE` w wywołaniu `MQCLOSE`, kolejka może nie być pusta. Jeśli jakiegokolwiek publikacje z menedżera kolejek nie zostały jeszcze zatwierdzone po usunięciu kolejki, wywołanie `MQCLOSE` jest wydawane przez kod powrotu `MQRC_Q_NOT_EMPTY`. Aplikacja może unikać tego problemu przez spanie i ponowne wywołanie wywołania `MQCLOSE` od czasu do czasu.

Właściwości

< komenda> (MQPSC_COMMAND)

Wartością jest `DeregSub` (`MQPSC_DEREGISTER_SUBSKRYBENTA`).

Ta właściwość musi być określona.

< Topic> (MQPSC_TOPIC)

Wartość jest łańcuchem, który zawiera temat, który ma zostać wyrejestrowany.

Ta właściwość może opcjonalnie zostać powtórzona, jeśli wiele tematów ma zostać wyrejestrowywanych. Można go pominąć, jeśli wartość `DeregAll` jest określona w pliku `<RegOpt>`.

Określone tematy mogą być podzbiorem tych tematów, które są zarejestrowane, jeśli subskrybent chce zachować subskrypcje w innych tematach. Znaki wieloznaczne są dozwolone, ale łańcuch

tematu, który zawiera znaki wieloznaczne, musi być dokładnie zgodny z odpowiednim łańcuchem podanym w komunikacie komendy **Deregister Subscriber**.

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

Wartość jest łańcuchem, który określa punkt subskrypcji, z którego subskrypcja ma zostać odłączona.

Ta właściwość nie może być powtarzana. Można go pominąć, jeśli zostanie podana wartość < Topic>, lub jeśli wartość DeregAll jest określona w pliku <RegOpt>. Jeśli ta właściwość zostanie pominięta, wykonywane są następujące czynności:

- Jeśli **nie** zostanie określona wartość DeregAll, subskrypcje zgodne z właściwością < Topic> (i właściwość < Filter>, jeśli są obecne) zostaną wyrejestrowywane z domyślnego punktu subskrypcji.
- Jeśli zostanie określona wartość DeregAll, wszystkie subskrypcje (zgodne z właściwościami < Topic> i < Filter>, jeśli są obecne) zostaną wyrejestrowywane ze wszystkich punktów subskrypcji.

Należy pamiętać, że nie można jawnie określić domyślnego punktu subskrypcji. Oznacza to, że nie ma możliwości wyrejestrowania wszystkich subskrypcji tylko z tego punktu subskrypcji. Należy określić tematy.

<SubIdentity> (MQPSC_SUBSCRIPTION_IDENTITY)

Jest to łańcuch o zmiennej długości o maksymalnej długości 64 znaków. Jest on używany do reprezentowania aplikacji z zainteresowaniem w subskrypcji. Menedżer kolejek przechowuje zestaw tożsamości subskrybenta dla każdej subskrypcji. Każda subskrypcja może pozwolić na to, aby jego tożsamość była tylko pojedynczą tożsamością lub nieograniczoną liczbą tożsamości.

Jeśli element SubIdentity znajduje się w zestawie tożsamości dla subskrypcji, to jest on usuwany z zestawu. Jeśli w wyniku tego zestaw tożsamości stanie się pusty, subskrypcja zostanie usunięta z menedżera kolejek, chyba że jako wartość właściwości RegOpt została określona wartość LeaveOnly. Jeśli zestaw tożsamości nadal zawiera inne tożsamości, wówczas subskrypcja nie zostanie usunięta z menedżera kolejek, a przepływ publikacji nie zostanie przerwany.

Jeśli zostanie podana wartość SubIdentity, ale element SubIdentity nie znajduje się w zestawie tożsamości dla subskrypcji, komenda **Deregister Subscriber** nie powiedzie się i zostanie zwrócony kod powrotu **MQRCCF_SUB_IDENTITY_ERROR**.

< Filter> (MQPSC_FILTER)

Wartością jest łańcuch określający filtr, który ma zostać wyrejestrowany. Musi ona być dokładnie zgodna z filtrem subskrypcji, który został wcześniej zarejestrowany, w tym przypadku i ze wszystkich obszarów.

Ta właściwość może opcjonalnie zostać powtórzona, jeśli ma zostać wyrejestrowany więcej niż jeden filtr. Można go pominąć, jeśli zostanie podana wartość < Topic>, lub jeśli wartość DeregAll jest określona w pliku <RegOpt>.

Określone filtry mogą być podzbiorem tych zarejestrowanych, jeśli subskrybent chce zachować subskrypcje dla innych filtrów.

<RegOpt> (MQPSC_REGISTRATION_OPTION)

Właściwość opcji rejestracji może przyjmować następujące wartości:

DeregAll

(MQPSC_DEREGISTER_ALL)

Wszystkie zgodne subskrypcje zarejestrowane dla tego subskrybenta mają zostać wyrejestrowywane.

Jeśli zostanie podana wartość DeregAll:

- Opcje < Topic>, <SubPoint> i < Filter> mogą zostać pominięte.
- Opcje < Topic> i < Filter> mogą być powtarzane, jeśli jest to wymagane.
- <SubPoint> nie może być powtórzony.

Jeśli **nie** zostanie określona opcja DeregAll, wykonaj następujące czynności:

- Wartość < Topic> musi być określona i może być powtórzona, jeśli jest wymagana.
- Opcje <SubPoint> i < Filter > mogą zostać pominięte.
- <SubPoint> nie może być powtórzony.
- < Filtr > może być powtórzony, jeśli jest to wymagane.

Jeśli zarówno tematy, jak i filtry są powtarzane, wszystkie subskrypcje zgodne ze wszystkimi kombinacjami obu tych kombinacji zostaną usunięte. Na przykład komenda **Deregister Subscriber**, która określa trzy tematy i trzy filtry, podejmie próbę usunięcia dziewięciu subskrypcji.

IdentyfikatorCorrelAs

(MQPSC_CORREL_ID_AS_IDENTITY)

Element CorrelId w deskrytorze komunikatu (MQMD), który nie może być zerowy, służy do identyfikowania subskrybenta. Musi być zgodny z identyfikatorem CorrelId używanym w oryginalnej subskrypcji.

FullResp

(MQPSC_FULL_RESPONSE)

Jeśli określono wartość FullResp, wszystkie atrybuty subskrypcji są zwracane w komunikacie odpowiedzi, jeśli ta komenda nie zakończy się niepowodzeniem.

Jeśli określona jest wartość FullResp, komenda DeregAll nie jest dozwolona w komendzie **Deregister Subscriber**. Nie jest również możliwe określenie wielu tematów. Wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_REG_OPTIONS_ERROR* w obu przypadkach.

LeaveOnly

(TYLKO MQPSC_LEAVE_ONLY)

Jeśli zostanie podana wartość SubIdentity, która znajduje się w zestawie tożsamości dla subskrypcji, element SubIdentity zostanie usunięty z zestawu tożsamości dla subskrypcji. Subskrypcja nie została usunięta z menedżera kolejek, nawet jeśli wynikowy zestaw tożsamości jest pusty. Jeśli wartość SubIdentity nie znajduje się w zestawie tożsamości, wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_SUB_IDENTITY_ERROR*.

Jeśli parametr LeaveOnly jest określony bez elementu SubIdentity, wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_REG_OPTIONS_ERROR*.

Jeśli nie zostaną określone ani LeaveOnly, ani SubIdentity, subskrypcja zostanie usunięta bez względu na zawartość zestawu tożsamości dla subskrypcji.

NONE

(MQPSC_NONE)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości opcji rejestracji. Jeśli w tym samym czasie zostaną podane inne opcje, opcja Brak zostanie zignorowana.

VariableUserId

(ID_UŻYTKOWNIKA_WYWOŁANIA MQPSC_VARIABLE_USER_ID)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid), nie jest on ograniczony do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnej wiadomości rejestracyjnej z tożsamością subskrybenta, a od tego czasu uniemożliwia innym użytkownikom korzystanie z tej tożsamości. Jeśli nowy subskrybent podejmie próbę użycia tej samej tożsamości, zwracany jest kod powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Każdy użytkownik może zmodyfikować lub wyrejestrować subskrypcję, jeśli mają odpowiednie uprawnienia, unikając w ten sposób sprawdzenia, czy identyfikator użytkownika musi być zgodny z identyfikatorem oryginalnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli subskrypcja, która ma zostać wyrejestrowana, ma ustawioną wartość `VariableUserId`, musi być ustawiona w czasie wyrejestrowywania, aby wskazać, która subskrypcja jest wyrejestrowana. W przeciwnym razie do identyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Deregister Subscriber**. Wartość ta jest przestawiana wraz z innymi identyfikatorami subskrybenta, jeśli została podana nazwa subskrypcji.

Wartość domyślna, jeśli ta właściwość jest pominięta, oznacza, że nie są ustawione żadne opcje rejestracji.

<QMgrName> (MQPSC_Q_MGR_NAME)

Wartością jest nazwa menedżera kolejek dla kolejki subskrybenta. Musi być ona zgodna z `QMgrName` użytym w oryginalnej subskrypcji.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa Menedżer kolejek produktu `ReplyTo` w deskrytorze komunikatu (`MQMD`). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest nazwa menedżera kolejek.

< nazwa_QName> (NAZWA_Q_ZMATERIALIZOWANIA_Z_MQ_MQ)

Wartość jest nazwą kolejki subskrybenta. Musi być ona zgodna z nazwą `QName` używaną w oryginalnej subskrypcji.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQ` w deskrytorze komunikatu (`MQMD`), która nie może być pusta.

<SubName> (MQPSC_SUBSCRIPTION_NAME)

Jeśli w komendzie **Deregister Subscriber** zostanie podana wartość `SubName`, wartość `SubName` ma pierwszeństwo przed wszystkimi innymi polami identyfikatora, z wyjątkiem identyfikatora użytkownika, chyba że dla subskrypcji zostanie ustawiona wartość `VariableUserId`. Jeśli parametr `VariableUserId` nie jest ustawiony, komenda **Deregister Subscriber** powiedzie się tylko wtedy, gdy identyfikator użytkownika komunikatu komendy jest zgodny z identyfikatorem subskrypcji, jeśli nie, wykonanie komendy kończy się niepowodzeniem z kodem powrotu `MQRCCF_DUPLICATE_IDENTITY`.

Jeśli istnieje subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, ale nie ma opcji `SubName`, komenda **Deregister Subscriber** nie powiedzie się z kodem powrotu `MQRCCF_SUB_NAME_ERROR`. Jeśli podjęta zostanie próba wyrejestrowania subskrypcji, która ma `SubName`, za pomocą komunikatu komendy zgodnego z tradycyjną tożsamością, ale bez określonej opcji `SubName`, komenda zakończy się powodzeniem.

<SubUser> (MQPSC_SUBSCRIPTION_USER_DATA)

Jest to łańcuch tekstowy o zmiennej długości. Wartość jest zapisywana przez menedżer kolejek z subskrypcją, ale nie ma wpływu na dostarczenie publikacji do subskrybenta. Wartość tę można zmienić, rejestrując ją w tej samej subskrypcji o nową wartość. Ten atrybut służy do korzystania z aplikacji.

`SubUserDane` są zwracane w informacjach dotyczących Metatopic (`MQCACF_REG_SUB_USER_DATA`) dla subskrypcji, jeśli dane `SubUsers` są obecne.

Przykład

Poniżej znajduje się przykład danych `NameValueData` dla komunikatu komendy **Deregister Subscriber**. W tym przykładzie przykładowa aplikacja wyrejestrowuje swoją subskrypcję tematów, które zawierają najnowszy wynik dla wszystkich dopasowań. Tożsamość subskrybenta, w tym identyfikator `CorrelId`, jest pobierana z wartości domyślnych w strukturze `MQMD`.

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publikuj komunikat

Komunikat komendy **Publish** jest umieszczany w kolejce lub z menedżera kolejek do subskrybenta w celu publikowania informacji dotyczących określonego tematu lub tematów.

Wymagane jest uprawnienie do umieszczenia komunikatu w kolejce i uprawnienia do publikowania informacji na określony temat lub tematy.

Jeśli użytkownik ma uprawnienia do publikowania informacji o niektórych, ale nie wszystkich, tematach, tylko te tematy są używane do publikowania. Odpowiedź ostrzeżenia wskazuje, które tematy nie są używane do publikowania.

Jeśli subskrybent ma jakiegokolwiek zgodne subskrypcje, menedżer kolejek przekazuje komunikat **Publish** do kolejek subskrybenta zdefiniowanych w odpowiednich komunikatach komend produktu **Register Subscriber**.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD) wymaganych podczas wysyłania komunikatu komendy do menedżera kolejek i używane w przypadku, gdy menedżer kolejek przekazuje publikację do subskrybenta, zawiera sekcja [Komunikat odpowiedzi menedżera kolejek](#).

Menedżer kolejek przekazuje komunikat **Publish** do innych menedżerów kolejek w sieci, które mają zgodne subskrypcje, o ile nie jest to publikacja lokalna.

Dane publikacji, jeśli są dostępne, są zawarte w treści wiadomości. Dane mogą być opisane w folderze <mcd> w polu NameValueData w nagłówku MQRFH2.

Właściwości

< komenda> (MQPSC_COMMAND)

Wartość ta wynosi Publikuj (MQPSC_PUBLISH).

Ta właściwość musi być określona.

< Topic> (MQPSC_TOPIC)

Wartość jest łańcuchem zawierającym temat, który kategoryzuje tę publikację. Znaki zastępcze nie są dozwolone.

Należy dodać temat do listy nazw SYSTEM.QPUBSUB.QUEUE.NAMELIST, patrz sekcja [Dodawanie strumienia](#), aby uzyskać instrukcje dotyczące sposobu wykonania tego zadania.

Ta właściwość musi być określona i może być opcjonalnie powtórzona w razie potrzeby w przypadku wielu tematów.

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

Punkt subskrypcji, w którym publikowana jest publikacja.

W przypadku produktu WebSphere Event Broker V6 wartość właściwości <SubPoint> jest wartością atrybutu Punkt subskrypcji węzła Publication obsługującego publikowanie.

W programie IBM MQ V7.0.1 wartość właściwości <SubPoint> musi być zgodna z nazwą punktu subskrypcji. Patrz sekcja [Dodawanie punktu subskrypcji](#).

<PubOpt> (MQPSC_PUBLICATION_OPTION)

Właściwość opcji publikacji może przyjmować następujące wartości:

RetainPub

(MQPSC_RETAIN_PUB)

Menedżer kolejek ma zachować kopię publikacji. Jeśli ta opcja nie zostanie ustawiona, publikacja zostanie usunięta, gdy tylko menedżer kolejek wystąpi publikację do wszystkich jej bieżących subskrybentów.

IsRetainedPub

(MQPSC_IS_RETAINED_PUB)

(Może być ustawiona tylko przez menedżer kolejek). Ta publikacja została zachowana przez menedżera kolejek. Menedżer kolejek ustawia tę opcję w celu powiadomienia subskrybenta

o tym, że ta publikacja została opublikowana wcześniej i została zachowana pod warunkiem, że subskrypcja została zarejestrowana przy użyciu opcji `InformIfRetained` (Nieformalne informacje o tym produkcie). Jest ona ustawiana tylko w odpowiedzi na komunikat komendy `Register Subscriber` lub `Request Update`. Zachowane publikacje, które są wysyłane bezpośrednio do subskrybentów, nie mają tego zestawu opcji.

Lokalna

(*MQPSC_LOCAL*)

Ta opcja informuje menedżera kolejek o tym, że ta publikacja nie może być wysyłana do innych menedżerów kolejek. Wszyscy subskrybenci, którzy zarejestrowali się w tym menedżerze kolejek, otrzymują tę publikację, jeśli mają zgodne subskrypcje.

Tylko OtherSubs

(*TYLKO MQPSC_OTHER_SUBS_ONLY*)

Ta opcja umożliwia prostsze przetwarzanie aplikacji typu konferencyjnego, w przypadku których publikator jest również subskrybentem tego samego tematu. Informuje ona menedżera kolejek, aby nie wysyłała publikacji do kolejki subskrybenta publikatora, nawet jeśli ma zgodną subskrypcję. Kolejka subskrybenta publikatora składa się z jej `QMGrName`, `QNamei` opcjonalnego elementu `CorrelId`, zgodnie z opisem na poniższej liście.

IdentyfikatorCorrelAs

(*MQPSC_CORREL_ID_AS_IDENTITY*)

Element `CorrelId` w strukturze `MQMD` (który nie może być zerem) należy do kolejki subskrybenta publikatora, w aplikacjach, w których publikator jest również subskrybentem.

NONE

(*MQPSC_NONE*)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości opcji publikacji. Jeśli w tym samym czasie zostaną podane inne opcje, opcja `Brak` zostanie zignorowana.

Użytkownik może mieć więcej niż jedną opcję publikowania, wprowadzając dodatkowe elementy `<PubOpt>`.

Domyślnie, jeśli ta właściwość jest pominięta, to nie są ustawione żadne opcje publikowania.

<PubTime> (MQPSC_PUBLISH_TIMESTAMP)

Wartość jest opcjonalnym znacznikiem czasu publikacji ustawionym przez publikator. Ma on 16 znaków długości w formacie:

```
YYYYMMDDHHMSSSTH
```

przy użyciu czasu uniwersalnego. Informacje te nie są sprawdzane przez menedżera kolejek przed wysłaniem do subskrybentów.

<SeqNum> (MQPSC_SEQUENCE_NUMBER)

Wartość jest opcjonalnym numerem kolejnym ustawionym przez publikator.

Musi być ona zwiększana o 1 z każdą publikacją. Nie jest to jednak sprawdzane przez menedżera kolejek, który przesyła tylko te informacje do subskrybentów.

Jeśli publikacje w tym samym temacie są publikowane w różnych połączonych ze sobą menedżerach kolejek, publikatory muszą mieć pewność, że numery kolejne, jeśli są używane, mają znaczenie.

<QMGrName> (MQPSC_Q_MGR_NAME)

Wartość jest łańcuchem zawierającym nazwę menedżera kolejek dla kolejki subskrybenta publikatora, w aplikacjach, w których publikator jest również subskrybentem (patrz `OtherSubsTylko`).

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `Menedżer kolejek produktu ReplyTo` w deskrytorze komunikatu (`MQMD`). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest nazwa menedżera kolejek.

< nazwa_QName> (NAZWA_Q_ZMATERIALIZOWANIA_Z_MQ_MQ)

Wartość jest łańcuchem zawierającym nazwę kolejki subskrybenta publikatora, w aplikacjach, w których publikator jest również subskrybentem (patrz `OtherSubsTyłko`).

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQ` w deskrytorze komunikatu (MQMD), która nie może być pusta, jeśli ustawiona jest wartość `OtherSubsOnly`.

Przykład

Poniżej przedstawiono kilka przykładów wartości *NameValueData* (Dane nazwy) dla komunikatu komendy **Publish**.

Pierwszy przykład dotyczy publikacji wystanej przez symulator zgodności w przykładowej aplikacji w celu wskazania, że zgodność została rozpoczęta.

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

Drugi przykład dotyczy zachowanej publikacji. Najnowszy wynik w uzgodnieniu między Team1 i Team2 jest opublikowany.

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Rejestrowanie komunikatu subskrybenta

Komunikat komendy **Register Subscriber** jest wysyłany do menedżera kolejek przez subskrybent lub przez inną aplikację w imieniu subskrybenta, aby wskazać, że chce subskrybować jeden lub więcej tematów w punkcie subskrypcji. Można również określić filtr treści komunikatu.

W wyrażeniach filtru publikowania/subskrypcji nawiasy zagnieżdżające powodują spadek wydajności w sposób wykładniczy. Unikaj zagnieżdżania nawiasów do głębokości większej niż około 6.

Komunikat zostanie wysłany do systemu `SYSTEM.BROKER.CONTROL.QUEUE`, która jest kolejką sterującą menedżera kolejek. Wymagane jest uprawnienie do umieszczenia komunikatu w tej kolejce, oprócz uprawnień dostępu (ustawionych przez administratora systemu menedżera kolejek) dla tematu lub tematów w subskrypcji.

Jeśli użytkownik ma uprawnienia do niektórych, ale nie wszystkich, rejestrowane są tylko te, które mają uprawnienia; ostrzeżenie to wskazuje te, które nie są zarejestrowane.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD), które są wymagane podczas wysyłania komunikatu komendy do menedżera kolejek, zawiera sekcja [“Ustawienia MQMD w komunikatach komend do menedżera kolejek”](#) na stronie 909.

Jeśli odpowiedź na kolejkę jest tymczasową kolejką dynamiczną, subskrypcja zostanie automatycznie wyrejestrowana przez menedżer kolejek po zamknięciu kolejki.

Właściwości

< komenda> (MQPSC_COMMAND)

Wartością jest `RegSub` (`MQPSC_REGISTER_SUBSKRYBENTA`). Ta właściwość musi być określona.

< Topic> (MQPSC_TOPIC)

Temat, dla którego subskrybent chce otrzymywać publikacje. Znaki wieloznaczne mogą być określone jako część tematu.

Jeśli do sprawdzenia subskrypcji utworzonej w ten sposób zostanie użyta komenda MQSC **display sub**, to wartość znacznika `< Topic>` jest wyświetlana jako właściwość `TOPICSTR` subskrypcji.

Ta właściwość jest wymagana i może być opcjonalnie powtórzona w razie potrzeby w przypadku wielu tematów.

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

Wartość jest punktem subskrypcji, do którego przyłączona jest subskrypcja.

Jeśli ta właściwość zostanie pominięta, zostanie użyty domyślny punkt subskrypcji.

W programie WebSphere Event Broker V6 wartość właściwości <SubPoint> musi być zgodna z wartością atrybutu Punkt subskrypcji węzłów Publication, które są subskrybowane.

W programie IBM MQ V7.0.1 wartość właściwości <SubPoint> musi być zgodna z nazwą punktu subskrypcji. Patrz sekcja [Dodawanie punktu subskrypcji](#).

< Filt r > (MQPSC_FILTER)

Wartość jest wyrażeniem SQL używanym jako filtr w treści komunikatów publikacji. Jeśli publikacja w określonym temacie jest zgodna z filtrem, zostanie ona wysłana do subskrybenta. Ta właściwość odpowiada łańcuchowi wyboru, który jest używany w wywołaniach MQSUB i MQOPEN. Więcej informacji na ten temat zawiera sekcja [Wybieranie treści komunikatu](#)

Jeśli ta właściwość zostanie pominięta, filtrowanie treści nie będzie mieć miejsca.

<RegOpt> (MQPSC_REGISTRATION_OPTION)

Ta właściwość opcji rejestracji może przyjmować następujące wartości:

AddName

(MQPSC_ADD_NAME)

Jeśli określono dla istniejącej subskrypcji, która jest zgodna z tradycyjną tożsamością tej komendy rejestru subskrypcji, ale bez bieżącej wartości SubName , do subskrypcji zostanie dodana wartość SubName określona w tej komendzie.

Jeśli określono wartość AddName , pole SubName jest obowiązkowe. W przeciwnym razie zwracana jest wartość MQRCCF_REG_OPTIONS_ERROR.

IdentyfikatorCorrelAs

(MQPSC_CORREL_ID_AS_IDENTITY)

Element CorrelId w deskrytorze komunikatu (MQMD) jest używany podczas wysyłania zgodnych publikacji do kolejki subskrybenta. Wartość CorrelId nie może być równa zero,

FullResp

(MQPSC_FULL_RESPONSE)

Po określeniu wszystkich atrybutów subskrypcji w komunikacie odpowiedzi, jeśli komenda nie zakończy się niepowodzeniem.

FullResp jest poprawna tylko w przypadku, gdy komunikat komendy odnosi się do pojedynczej subskrypcji. Z tego powodu w komendzie dozwolony jest tylko jeden temat. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu MQRCCF_REG_OPTIONS_ERROR.

InformIfRet

(MQPSC_INFORM_IF_ZACHOWANE)

Menedżer kolejek informuje subskrybenta, jeśli publikacja jest zachowywana, gdy wysyła komunikat publikowania w odpowiedzi na komunikat komendy **Register Subscriber** lub **Request Update** . Menedżer kolejek wykonuje to działanie, dołączając opcję publikowania IsRetainedPub w komunikacie.

JoinExcl

(MQPSC_JOIN_EXCLUSIVE)

Ta opcja wskazuje, że określony element SubIdentity powinien zostać dodany jako wyłączny element zestawu tożsamości dla subskrypcji, a także, że do zestawu nie można dodać żadnych innych tożsamości.

Jeśli tożsamość już dołączyła do 'shared' i jest jedynym wpisem w zestawie, to zestaw zostanie zmieniony na blokadę na wyłączność posiadaną przez tę tożsamość. W przeciwnym razie, jeśli subskrypcja ma obecnie inne tożsamości w zestawie tożsamości (z dostępem współużytkowanym), wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu `MQRCCF_SUBSCRIPTION_IN_USE`.

JoinShared

(`MQPSC_JOIN_SHARED`)

Ta opcja wskazuje, że określona wartość `SubIdentity` powinna zostać dodana do zestawu tożsamości dla subskrypcji.

Jeśli subskrypcja jest obecnie zablokowana wyłącznie (za pomocą opcji `JoinExcl`), wykonanie komendy kończy się niepowodzeniem z kodem powrotu `MQRCCF_SUBSCRIPTION_LOCKED`, chyba że tożsamość, która ma zablokowaną subskrypcję, jest taką samą tożsamością, jak ta w komunikacie komendy. W tym przypadku blokada jest automatycznie modyfikowana do blokady ze współużytkiem.

Lokalna

(`MQPSC_LOCAL`)

Subskrypcja jest lokalna i nie jest dystrybuowana do innych menedżerów kolejek w sieci. Publikacje dokonane w innych menedżerach kolejek nie są dostarczane do tego subskrybenta, chyba że ma również odpowiednią subskrypcję globalną.

TylkoNewPubs

(`MQPSC_NEW_PUBS_ONLY`)

Zachowane publikacje, które istnieją w momencie rejestracji subskrypcji, nie są wysyłane do subskrybenta; wysyłane są tylko nowe publikacje.

Jeśli subskrybent jest ponownie rejestrowany i zmienia tę opcję, tak aby nie była już ustawiona, może zostać ponownie wysłana publikacja, która została już wysłana do niej.

NoAlter

(`MQPSC_NO_ALTER`)

Atrybuty istniejącej zgodnej subskrypcji nie zostały zmienione.

Gdy tworzona jest subskrypcja, ta opcja jest ignorowana. Wszystkie pozostałe opcje mają zastosowanie do nowej subskrypcji.

Jeśli w polu `SubIdentity` znajduje się również jedna z opcji łączenia (`JoinExcl` lub `JoinShared`) określono, tożsamość jest dodawana do zestawu tożsamości bez względu na to, czy została określona opcja `NoAlter`.

NONE

(`MQPSC_NONE`)

Wszystkie opcje rejestracji przyjmują wartości domyślne.

Jeśli subskrybent jest już zarejestrowany, jego opcje są resetowane do wartości domyślnych (należy zauważyć, że *nie* ma to samo wpływu na pominięcie właściwości opcji rejestracji), a utrata ważności subskrypcji jest aktualizowana na podstawie deskryptora MQMD komunikatu produktu **Register Subscriber**.

Jeśli w tym samym czasie zostaną podane inne opcje rejestracji, opcja `Brak` zostanie zignorowana.

NonPers

(`MQPSC_NON_PERSISTENT`)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta jako komunikaty nietrwałe.

Pers

(`MQPSC_PERSISTENT`)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta jako komunikaty trwałe.

PublikacjePersAs

(MQPSC_PERSISTENT_AS_PUBLISH)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta z trwałością określoną przez publikator. To jest zachowanie domyślne.

KolejkaPersAs

(MQPSC_PERSISTENT_AS_Q)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta z trwałością określoną w kolejce subskrybenta.

PubOnReqOnly

(MQPSC_PUB_ON_REQUEST_ONLY)

Menedżer kolejek nie wysyła publikacji do subskrybenta, z wyjątkiem odpowiedzi na komunikat komendy **Request Update**.

VariableUserId

(ID_UŻYTKOWNIKA_WYWOŁANIA MQPSC_VARIABLE_USER_ID)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid), nie jest on ograniczony do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnej wiadomości rejestracyjnej z tożsamością subskrybenta, a od tego czasu uniemożliwia innym użytkownikom korzystanie z tej tożsamości. Jeśli nowy subskrybent podejmie próbę użycia tej samej tożsamości *MQRCCF_DUPLICATE_SUBSCRIPTION*, zostanie zwrócona.

Dzięki temu dowolny użytkownik może zmodyfikować lub wyrejestrować subskrypcję, jeśli użytkownik ma odpowiednie uprawnienia. Dlatego nie ma potrzeby sprawdzania, czy identyfikator użytkownika jest zgodny z identyfikatorem oryginalnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli subskrypcja komendy **Request Update** ma ustawioną wartość *VariableUserId*, musi ona być ustawiona na czas aktualizacji żądania, aby wskazać, do której subskrypcji jest przywołana subskrypcja. W przeciwnym razie do identyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Request Update**. Wartość ta jest przestawiana wraz z innymi identyfikatorami subskrybenta, jeśli została podana nazwa subskrypcji.

Jeśli komunikat komendy **Register Subscriber** bez tego zestawu opcji odwołuje się do istniejącej subskrypcji, która ma ten zestaw opcji, ta opcja zostanie usunięta z tej subskrypcji, a identyfikator użytkownika subskrypcji zostanie teraz naprawiony. Jeśli istnieje już subskrybent, który ma taką samą tożsamość (kolejkę, menedżer kolejek i identyfikator korelacji), ale z innym powiązaniem identyfikatorem użytkownika, wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_IDENTITY*, ponieważ może istnieć tylko jeden identyfikator użytkownika powiązany z tożsamością subskrybenta.

Jeśli właściwość opcji rejestracji zostanie pominięta, a subskrybent jest już zarejestrowany, jej opcje rejestracji nie zostaną zmienione, a utrata ważności subskrypcji zostanie zaktualizowana z deskryptora MQMD komunikatu produktu **Register Subscriber**.

Jeśli subskrybent nie jest jeszcze zarejestrowany, zostanie utworzona nowa subskrypcja ze wszystkimi opcjami rejestracji, w których zostaną podane wartości domyślne.

Wartości domyślne to *PersAsPub* i nie są ustawione żadne inne opcje.

<QMgrName> (MQPSC_Q_MGR_NAME)

Wartością jest nazwa menedżera kolejek dla kolejki subskrybenta, do której są wysyłane zgodne publikacje przez menedżer kolejek.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa Menedżer kolejek produktu *ReplyTo* w deskrypcji komunikatu (MQMD). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest *QMgrName* menedżera kolejek.

< nazwa_QName> (NAZWA_Q_ZMATERIALIZOWANIA_Z_MQ_MQ)

Wartość jest nazwą kolejki subskrybenta, do której są wysyłane zgodne publikacje przez menedżera kolejek.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQ w deskrypcji komunikatu (MQMD), która w tym przypadku nie może być pusta.

Jeśli kolejka jest krótkotrwałą kolejką dynamiczną, nietrwałe dostarczanie publikacji (NonPers) musi być określony we właściwości <RegOpt>.

Jeśli kolejka jest tymczasową kolejką dynamiczną, subskrypcja jest automatycznie wyrejestrowana przez menedżera kolejek, gdy kolejka jest zamknięta.

<SubName> (MQPSC_SUBSCRIPTION_NAME)

Jest to nazwa nadana konkretnej subskrypcji. Można go używać zamiast menedżera kolejek, kolejki i opcjonalnego identyfikatora correlId w celu odwołania się do subskrypcji.

Jeśli subskrypcja istnieje już z tym produktem **SubName**, wszystkie inne atrybuty subskrypcji (Temat, QMgrName, Nazwa QName, CorrelId, UserId, RegOpts, UserSubDane i Utrata ważności) są nadpisywane z atrybutami, jeśli są one określone, które są przekazywane w nowym komunikacie komendy Register Subscriber. Jeśli jednak produkt **SubName** zostanie użyty bez określonego pola nazwy QName, a w nagłówku MQMD zostanie podany parametr ReplyToQ, to kolejka subskrybenta zostanie zmieniona na ReplyToQ.

Jeśli subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, już istnieje, ale nie ma **SubName**, komenda rejestracji kończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*, o ile nie zostanie podana opcja **AddName**.

W przypadku próby zmiany istniejącej subskrypcji nazwanej za pomocą innej komendy Register Subscriber, która określa ten sam **SubName**, oraz wartości tematów Temat, QMgrName, QName i CorrelId w nowej komendzie są zgodne z inną istniejącą subskrypcją, z SubName lub bez niego, wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*. Uniemożliwia to dwie nazwy subskrypcji odnoszące się do tej samej subskrypcji.

<SubIdentity> (MQPSC_SUBSCRIPTION_IDENTITY)

Ten łańcuch jest używany do reprezentowania aplikacji z zainteresowaniem w subskrypcji. Jest to łańcuch znaków o zmiennej długości o maksymalnej długości 64 znaków i jest opcjonalny. Menedżer kolejek przechowuje zestaw tożsamości subskrybenta dla każdej subskrypcji. Każda subskrypcja może pozwolić, aby jej zestaw tożsamości zawierał tylko jedną tożsamość, lub nieograniczoną liczbę tożsamości (patrz opcje **JoinShared** i **JoinExcl**).

Komenda subskrypcji, która określa opcję **JoinShared** lub **JoinExcl**, dodaje **SubIdentity** do zestawu tożsamości subskrypcji, jeśli jeszcze nie istnieje i jeśli istniejący zbiór tożsamości zezwala na takie działanie; oznacza to, że żaden inny subskrybent nie dołączył wyłącznie lub zestaw tożsamości nie jest pusty.

Każda zmiana atrybutów subskrypcji w wyniku działania komendy Register Subscriber, w której określony jest parametr **SubIdentity**, tylko wtedy, gdy będzie jedynym elementem zestawu tożsamości dla tej subskrypcji. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_SUBSCRIPTION_IN_USE*. Zapobiega to zmianie atrybutów subskrypcji bez informacji o innych zainteresowanych subskrybentach.

Jeśli zostanie określony łańcuch znaków o długości większej niż 64 znaki, wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_SUB_IDENTITY_ERROR*.

<SubUser> (MQPSC_SUBSCRIPTION_USER_DATA)

Jest to łańcuch tekstowy o zmiennej długości. Wartość ta jest zapisywana przez menedżera kolejek w subskrypcji, ale nie ma wpływu na dostarczanie publikacji do subskrybenta. Wartość tę można zmienić, rejestrując ją w tej samej subskrypcji o nową wartość. Ten atrybut jest używany do korzystania z aplikacji.

SubUserDane jest zwracany w informacjach dotyczących metatematu (*MQCACF_REG_SUB_USER_DATA*) w przypadku subskrypcji, jeśli istnieje.

Jeśli zostanie podana więcej niż jedna z wartości opcji rejestracji `NonPers`, `PersAsPub`, `PersAsQueue`, and `Pers`, zostanie użyta tylko ostatnia wartość. Nie można łączyć tych opcji w ramach subskrypcji indywidualnej.

Przykład

Poniżej znajduje się przykład danych `NameValueData` dla komunikatu komendy **Register Subscriber**. W przykładowej aplikacji usługa wyników korzysta z tego komunikatu w celu zarejestrowania subskrypcji w tematach zawierających najnowsze wyniki we wszystkich dopasach, przy czym zestaw opcji "Trwałe jako publikowanie" jest ustawiony. Tożsamość subskrybenta, w tym identyfikator `CorrelId`, jest pobierana z wartości domyślnych w strukturze `MQMD`.

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Komunikat o aktualizacji żądania

Komunikat komendy **Request Update** jest wysyłany z subskrybenta do menedżera kolejek w celu żądania bieżących zachowanych publikacji dla określonego tematu i punktu subskrypcji, które są zgodne z podanym filtrem (opcjonalnym).

Ten komunikat jest wysyłany do systemu `SYSTEM.BROKER.CONTROL.QUEUE`, kolejka sterująca menedżera kolejek. Wymagane jest uprawnienie do umieszczenia komunikatu w tej kolejce, a także uprawnienie dostępu do tematu w aktualizacji żądania. Jest to ustawiane przez administratora systemu menedżera kolejek.

Ta komenda jest zwykle używana, jeśli subskrybent określił opcję `PubOnReqOnly`, gdy została zarejestrowana. Jeśli menedżer kolejek ma jakiegokolwiek zgodne zachowane publikacje, są one wysyłane do subskrybenta. Jeśli menedżer kolejek nie ma zgodnych zachowywanych publikacji, żądanie nie powiedzie się i zostanie zwrócony kod powrotu `MQRCCF_NO_RETAINED_MSG`. Żadający musiał wcześniej zarejestrować subskrypcję o tym samym temacie, `SubPoint` wartości filtru.

Właściwości

< komenda> (MQPSC_COMMAND)

Wartość jest ustawiona na `ReqUpdate` (`MQPSC_REQUEST_UPDATE`). Ta właściwość musi być określona.

< Topic> (MQPSC_TOPIC)

Wartością jest temat, do którego żąda się subskrybent; dozwolone są znaki wieloznaczne.

Ta właściwość musi być określona, ale tylko jedno wystąpienie jest dozwolone w tym komunikacie.

<SubPoint> (MQPSC_SUBSCRIPTION_POINT)

Wartość jest punktem subskrypcji, do którego przyłączona jest subskrypcja.

Jeśli ta właściwość zostanie pominięta, zostanie użyty domyślny punkt subskrypcji.

< Filtr > (MQPSC_FILTER)

Wartość jest wyrażeniem ESQL, które jest używane jako filtr dla treści komunikatów publikacji. Jeśli publikacja w określonym temacie jest zgodna z filtrem, zostanie ona wysłana do subskrybenta.

Właściwość < Filter > powinna mieć taką samą wartość, jak wartość określona w pierwotnej subskrypcji, dla której teraz żąda się aktualizacji.

Jeśli ta właściwość zostanie pominięta, filtrowanie treści nie będzie mieć miejsca.

<RegOpt> (MQPSC_REGISTRATION_OPTION)

Właściwość opcji rejestracji może mieć następującą wartość:

IdentyfikatorCorrelAs

(MQPSC_CORREL_ID_AS_IDENTITY)

Identyfikator CorrelId w deskrytorze komunikatu (MQMD), który nie może być równy zero, jest używany podczas wysyłania zgodnych publikacji do kolejki subskrybenta.

NONE

(MQPSC_NONE)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości <RegOpt> . Jeśli w tym samym czasie zostaną podane inne opcje, opcja Brak zostanie zignorowana.

VariableUserId

(ID_UŻYTKOWNIKA_WYWOŁANIA MQPSC_VARIABLE_USER_ID)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid), nie jest on ograniczony do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnej wiadomości rejestracyjnej z tożsamością subskrybenta, a od tego czasu uniemożliwia innym użytkownikom korzystanie z tej tożsamości. Jeśli nowy subskrybent podejmie próbę użycia tej samej tożsamości, wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu MQRCCF_DUPLICATE_SUBSCRIPTION.

Dzięki temu każdy użytkownik może zmodyfikować lub wyrejestrować subskrypcję, gdy mają odpowiednie uprawnienia. Oznacza to, że nie ma potrzeby sprawdzania, czy identyfikator użytkownika jest zgodny z identyfikatorem oryginalnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika co oryginalna subskrypcja.

Jeśli subskrypcja komendy **Request Update** ma ustawioną wartość VariableUserId , musi ona być ustawiona na czas aktualizacji żądania, aby wskazać, do której subskrypcji jest przywołana subskrypcja. W przeciwnym razie do identyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Request Update** . Wartość ta jest przestawiana wraz z innymi identyfikatorami subskrybenta, jeśli została podana nazwa subskrypcji.

Wartość domyślna, jeśli ta właściwość jest pominięta, oznacza, że nie są ustawione żadne opcje rejestracji.

<QMgrName> (MQPSC_Q_MGR_NAME)

Wartość jest nazwą menedżera kolejek dla kolejki subskrybenta, do której pasująca zachowana publikacja jest wysyłana przez menedżer kolejek.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa Menedżer kolejek produktu ReplyTo w deskrytorze komunikatu (MQMD). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest QMgrNamemenedżera kolejek.

< nazwa_QName> (NAZWA_Q_ZMATERIALIZOWANIA_Z_MQ_MQ)

Wartość jest nazwą kolejki subskrybenta, do której pasująca zachowana publikacja jest wysyłana przez menedżer kolejek.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQ w deskrytorze komunikatu (MQMD), która w tym przypadku nie może być pusta.

<SubName> (MQPSC_SUBSCRIPTION_NAME)

Jest to nazwa nadana konkretnej subskrypcji. Jeśli określono w komendzie **Request Update** , wartość SubName ma pierwszeństwo przed wszystkimi innymi polami identyfikatora, z wyjątkiem identyfikatora użytkownika, chyba że VariableUserId jest ustawiony w samej subskrypcji. Jeśli parametr VariableUserId nie jest ustawiony, komenda *Request Update* zakończy się powodzeniem tylko wtedy, gdy identyfikator użytkownika w komunikacie komendy jest zgodny z identyfikatorem subskrypcji. Jeśli identyfikator użytkownika w komunikacie komendy nie jest zgodny z identyfikatorem subskrypcji, wykonanie komendy kończy się niepowodzeniem z kodem powrotu MQRCCF_DUPLICATE_IDENTITY.

Jeśli parametr `VariableUserId` jest ustawiony, a identyfikator użytkownika różni się od identyfikatora subskrypcji, komenda zakończy się powodzeniem, jeśli identyfikator użytkownika nowego komunikatu komendy ma uprawnienie do przeglądania kolejki strumienia i umieszczania w kolejce subskrybenta subskrypcji. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu `MQRCCF_NOT_AUTHORIZED`.

Jeśli istnieje subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, ale nie ma wartości `SubName`, komenda **Request Update** nie powiedzie się i zwrócony zostanie kod powrotu `MQRCCF_SUB_NAME_ERROR`.

Jeśli podejmowana jest próba żądania aktualizacji dla subskrypcji o nazwie `SubName` za pomocą komunikatu komendy zgodnego z tradycyjną tożsamością, ale nie określono parametru `SubName`, komenda zakończy się powodzeniem.

Przykład

Poniżej znajduje się przykład danych `NameValueData` dla komunikatu komendy **Request Update**. W przykładowej aplikacji usługa wyników używa tego komunikatu do żądania zachowanych publikacji, które zawierają najnowsze wyniki dla wszystkich zespołów. Tożsamość subskrybenta, w tym identyfikator `CorrelId`, jest pobierana z wartości domyślnych w strukturze `MQMD`.

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Komunikat odpowiedzi menedżera kolejek

Komunikat **Queue Manager Response** jest wysyłany z menedżera kolejek do kolejki `ReplyToQ` publikatora lub subskrybenta w celu wskazania powodzenia lub niepowodzenia komunikatu komendy odebranego przez menedżer kolejek, jeśli w deskrytorze komunikatu komendy określono, że odpowiedź jest wymagana.

Komunikat odpowiedzi znajduje się w polu `NameValueData` nagłówka `MQRFH2`, w folderze `<pscr>`.

W przypadku wystąpienia ostrzeżenia lub błędu komunikat odpowiedzi zawiera folder `<psc>` z poziomu komunikatu komendy oraz folder `<pscr>`. Dane komunikatu, jeśli istnieją, nie są zawarte w komunikacie odpowiedzi menedżera kolejek. W przypadku wystąpienia błędu żaden komunikat, który spowodował błąd, nie został przetworzony; w przypadku ostrzeżenia część komunikatu mogła zostać przetworzona pomyślnie.

Jeśli wystąpi błąd podczas wysyłania odpowiedzi:

- W przypadku komunikatów publikacji menedżer kolejek próbuje wysłać odpowiedź do kolejki niedostarczonych komunikatów serwera IBM MQ, jeśli operacja `MQPUT` nie powiedzie się. Pozwala to na wysyłanie publikacji do subskrybentów nawet wtedy, gdy odpowiedź nie może zostać wysłana z powrotem do publikatora.
- W przypadku innych komunikatów lub jeśli odpowiedź na publikację nie może zostać wysłana do kolejki niedostarczonych komunikatów, rejestrowany jest błąd, a komunikat komendy jest zwykle wycofany. To, czy dzieje się tak, zależy od sposobu skonfigurowania węzła `MQInput`.

Właściwości

< Completion> (MQPSCR_COMPLETION)

Kod zakończenia, który może przyjmować jedną z trzech wartości:

OK

Komenda została zakończona pomyślnie

ostrzeżenie

Komenda została zakończona, ale z ostrzeżeniem

błąd

Błąd komendy

< Response> (MQPSCR_RESPONSE)

Odpowiedź na komunikat komendy, jeśli ta komenda wygenerowała kod zakończenia warning (ostrzeżenie) lub error(błąd). Zawiera ona właściwość < Reason> i może zawierać inne właściwości, które wskazują przyczynę ostrzeżenia lub błędu.

W przypadku jednego lub większej liczby błędów istnieje tylko jeden folder odpowiedzi wskazujący przyczynę tylko pierwszego błędu. W przypadku jednego lub większej liczby ostrzeżeń istnieje folder odpowiedzi dla każdego ostrzeżenia.

< Reason> (MQPSCR_REASON)

Kod przyczyny kwalifikujący kod zakończenia, jeśli kod zakończenia ma wartość ostrzeżenie lub błąd. Jest on ustawiony na jeden z kodów błędów wymienionych w poniższym przykładzie. Właściwość < Reason> jest zawarta w folderze < Response> . Po kodzie przyczyny można śledzić dowolną poprawną właściwość z folderu <psc> (na przykład nazwę tematu), wskazującą przyczynę błędu lub ostrzeżenia. Jeśli dostajesz kod przyczyny? ???, sprawdź dane pod kątem poprawności, na przykład dopasowywanie nawiasów kątowych (< >).

Przykłady

Poniżej przedstawiono kilka przykładów wartości NameValueData w komunikacie **Queue Manager Response** . Pomyślna odpowiedź może być następująca:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Poniżej przedstawiono przykład odpowiedzi na awarię. Błąd ten jest błędem filtru. Pierwszy łańcuch NameValueData zawiera odpowiedź; druga zawiera oryginalną komendę.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
  ...
</psc>
```

Poniżej przedstawiono przykład odpowiedzi na ostrzeżenie (ze względu na nieautoryzowane tematy). Pierwszy łańcuch NameValueData zawiera odpowiedź, a drugi łańcuch NameValueData zawiera oryginalną komendę.

```
<pscr>
  <Completion>warning</Completion>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic1</Topic>
  </Reponse>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic2</Topic>
  </Reponse>
</pscr>

<psc>
  ...
  command message (to which
  the queue manager is responding)
```

</psc>

Kody przyczyny publikowania/subskrypcji

Te kody przyczyny mogą zostać zwrócone w polu Uzasadnienie w folderze <psc> odpowiedzi publikowania/subskrypcji. Wyświetlane są także stałe, które mogą być używane do reprezentowania tych kodów w językach programowania C lub C++.

Stałe MQRC_ wymagają pliku nagłówkowego IBM MQ cmqc.h. Stałe MQRCCF_ wymagają pliku nagłówkowego IBM MQ cmqcfc.h (oprócz plików *MQRCCF_FILTER_ERROR* i *MQRCCF_WRONG_USER*, które wymagają pliku nagłówkowego cmqpsc.h).

Kod przyczyny i tekst	Wyjaśnienie	Wystawiony przez
2336 MQRC_RFH_COMMAND_ERROR	Poprawne wartości dla pola <Command> w folderze <psc> to: RegSub, DeregSub, Publish, DeletePubi ReqUpdate. Wszystkie pozostałe wartości powodują wydanie tego kodu błędu.	Dowolna komenda
2337 MQRC_RFH_PARM_ERROR	Zarówno foldery <psc>, jak i <mcd> mają zestaw poprawnych parametrów, które można w nich określić. Sprawdź opisy tych folderów i upewnij się, że nie podano nieprawidłowych parametrów.	Dowolna komenda
2338 MQRC_RFH_DUPLICATE_PARM	Niektóre parametry (na przykład Temat) w folderze <psc> mogą być powtarzane, ale inne (na przykład Komenda) nie mogą być powtórzone. Upewnij się, że nie został zduplikowany parametr, który nie jest powtarzalny.	Dowolna komenda
2339 MQRC_RFH_PARM_MISSING	Niektóre parametry w folderach <psc> lub <mcd> są opcjonalne i można je pominąć; niektóre z nich są obowiązkowe i nie mogą być pomijane. Sprawdź, czy zostały uwzględnione wszystkie obowiązkowe parametry w folderach <psc> i <mcd>.	Dowolna komenda
2551 MQRC_SELECTION_NOT_AVAILABLE	Nie był dostępny żaden dostawca rozszerzonego wyboru komunikatów w celu określenia, które subskrybenci z określonym filtrem powinni otrzymać publikację.	Publikuj, Zarejestruj subskrybent i żądaj aktualizacji
	Brak dostępnego dostawcy rozszerzonego wyboru komunikatów do obsługi filtra określonego subskrybenta.	Zarejestruj subskrybent i żądanie aktualizacji
2554 BŁĄD MQRC_CONTENT_ERROR	Dostawca wyboru komunikatów rozszerzonych znalazł błąd w bieżącej lub zachowanej publikacji.	Publikuj i żądaj aktualizacji

Kod przyczyny i tekst	Wyjaśnienie	Wystawiony przez
3008 MQRCCF_COMMAND_NIE POWIODŁO SIĘ	Wystąpił błąd wewnętrzny, który uniemożliwił poprawne wykonanie komendy. Błąd może wystąpić, jeśli komenda została ponownie wydana. Dziennik zdarzeń systemowych dla menedżera kolejek zawiera informacje, które powinny być używane podczas zgłaszania problemu do produktu IBM.	Dowolna komenda
3072 BŁĄD MQRCCF_TOPIC_ERROR	Co najmniej jedna z wartości podanych dla parametru Temat jest niepoprawna. Sprawdź, czy wartości tematu są zgodne z podanymi ograniczeniami.	Dowolna komenda
3073 MQRCCF_NOT_ZAREJESTROWANY	Kombinacja SubPoint, Temat i Filtr, który został określony w komendzie DeregSub lub ReqUpdate , nie była kombinacją, z którą wcześniej zarejestrowano lub, w przypadku komendy DeregSub , jeśli została podana opcja DeregAll , jeden z właściwości SubPoint, Topic lub Filter nie został użyty do wyrejestrowania subskrypcji.	Wyrejestrowywanie subskrybentów i komend aktualizacji żądań
3074 Błąd MQRCCF_Q_MGR_NAME_ERROR	Podany menedżer kolejek był niepoprawny lub menedżer kolejek nie był dostępny lub nie istnieje.	Deregister subskrybent, publikowanie, rejestrowanie subskrybentów i komendy aktualizacji żądań
3076 MQRCCF_Q_NAME_ERROR-BŁĄD	Podana nazwa kolejki nie jest poprawna lub kolejka nie istnieje w określonym menedżerze kolejek.	Deregister subskrybent, publikowanie, rejestrowanie subskrybentów i komendy aktualizacji żądań
3077 MQRCCF_NO_RETAINED_MSG	Nie zostały zachowane komunikaty dla podanego tematu. Może to być błąd lub błąd, w zależności od projektu aplikacji.	Komenda Aktualizacja żądania

Kod przyczyny i tekst	Wyjaśnienie	Wystawiony przez
3079 MQRCCF_INCORRECT_Q	Komendy RegSub, DeregSubi ReqUpdate są zawsze wysyłane do systemu SYSTEM.BROKER.CONTROL.QUEUE (Kolejka) menedżera kolejek, dla którego są przeznaczone. Komendy publikowania i usuwania publikacji są wysyłane do kolejki wejściowej dla konkretnego przepływu komunikatów publikowania/subskrybowania, dla którego są przeznaczone. Jest to określone, gdy przepływ komunikatów jest zaprojektowany. Ten kod błędu jest zwracany, jeśli komenda została wysłana do niewłaściwej kolejki.	Dowolna komenda
3080 MQRCCF_CORREL_ID_ERROR-BŁĄD	Podano identyfikator CorrelAsjako jeden z parametrów RegOpt . Jednak pole CorrelId deskryptora MQMD nie zawiera poprawnego identyfikatora korelacji (to znaczy, że jest ustawiony na wartość MQCI_NONE).	Wyrejestrowywanie subskrybentów i rejestrowanie komend subskrybenta
3081 MQRCCF_NOT_AUTHORIZED (autoryzowany)	Brak autoryzacji do wykonania żądanej akcji. Ustawienia autoryzacji dla menedżera kolejek są obsługiwane przez administratora systemu przy użyciu edytora hierarchii tematów.	Publikuj i rejestruj komendy subskrybenta
3083 MQRCCF_REG_REG_OPTIONS_ERROR	Podano nierozpoznany parametr RegOpt w folderze <psc> , który zawiera komendę RegSub lub DeregSub .	Wyrejestrowywanie subskrybentów i rejestrowanie komend subskrybenta
3084 MQRCCF_PUB_OPTIONS_ERROR,	Określono nierozpoznany parametr PubOpt w folderze <psc> , który zawiera komendę publikowania.	Komenda publikowania
3087 MQRCCF_DEL_OPTIONS_ERROR,	Określono nierozpoznany parametr DelOpt w folderze <psc> , który zawiera komendę DeletePub .	Komenda Usunięcie publikacji
3150 MQRCCF_FILTER_ERROR	Wartość określona dla parametru Filter jest niepoprawna. Sprawdź sekcję, która opisuje poprawną składnię wyrażeń filtru, i upewnij się, że wyrażenie jest zgodne.	Wyrejestruj subskrybenta, zarejestruj subskrybent i komendy aktualizacji żądań
3151 MQRCCF_WRONG_USER	Subskrypcja, która jest zgodna z tą, która została określona, już istnieje. Jednak została zarejestrowana przez innego użytkownika. Subskrypcja może zostać zmieniona lub wyrejestrowana przez użytkownika, który pierwotnie go zarejestrował.	Wyrejestruj subskrybenta, zarejestruj subskrybent i komendy aktualizacji żądań

Kod przyczyny i tekst	Wyjaśnienie	Wystawiony przez
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Istnieje już zgodna subskrypcja o innej nazwie subskrypcji.	
3153 MQRCCF_SUB_NAME_ERROR	Format nazwy subskrypcji nie jest poprawny albo istnieje już zgodna subskrypcja bez nazwy subskrypcji.	
3154 MQRCCF_SUB_IDENTITY_ERROR-BŁĄD	Parametr tożsamości subskrypcji jest błędny. Podana wartość przekracza maksymalną dopuszczalną długość lub tożsamość subskrypcji nie jest aktualnie elementem zestawu tożsamości subskrypcji, a opcja rejestracji łączenia nie została określona.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Próba zmodyfikowania lub wyrejestrowania subskrypcji została podjęta przez element zestawu tożsamości, gdy nie był on jedynym elementem tego zestawu.	
3156 MQRCCF_SUBSCRIPTION_LOCKED	Subskrypcja jest obecnie zablokowana wyłącznie przez inną tożsamość.	
3157 MQRCCF_ALREADY_DOŁĄCZYŁ (a)	Podano opcję rejestracji łączenia, ale tożsamość subskrybenta była już elementem zestawu tożsamości subskrypcji.	

Ustawienia MQMD w komunikatach komend do menedżera kolejek

Aplikacje, które wysyłają komunikaty komend do menedżera kolejek, korzystają z następujących ustawień pól w deskrypcji komunikatu (MQMD). Pola, które są pozostawione jako wartość domyślna lub mogą być ustawione na dowolną poprawną wartość w zwykły sposób, nie są wyświetlane w tym miejscu.

Raport

Patrz MsgType i CorrelId.

MsgType

Parametr MsgType należy ustawić na wartość *MQMT_REQUEST* lub *MQMT_DATAGRAM*. Wartość *MQRC_MSG_TYPE_ERROR* zostanie zwrócona, jeśli parametr MsgType nie zostanie ustawiony na jedną z tych wartości.

Parametr MsgType powinien mieć wartość *MQMT_REQUEST* w przypadku komunikatu komendy, jeśli odpowiedź jest zawsze wymagana. Opcje MQRO_PAN i MQRO_NAN w polu Raport nie są istotne w tym przypadku.

Jeśli parametr MsgType jest ustawiony na wartość *MQMT_DATAGRAM*, odpowiedzi są zależne od ustawienia flag MQRO_PAN i MQRO_NAN w polu Raport :

- Tylko MQRO_PAN oznacza, że menedżer kolejek wysyła odpowiedź tylko wtedy, gdy ta komenda zakończy się powodzeniem.
- Tylko MQRO_NAN oznacza, że menedżer kolejek wysyła odpowiedź tylko wtedy, gdy wykonanie komendy nie powiedzie się.
- Jeśli komenda zakończy działanie z ostrzeżeniem, zostanie wysłana odpowiedź, jeśli ustawiona jest wartość MQRO_PAN lub MQRO_NAN.

- MQRO_PAN + MQRO_NAN oznacza, że menedżer kolejek wysyła odpowiedź, czy komenda zakończy się powodzeniem lub czy nie powiedzie się. Ma to ten sam efekt z perspektywy menedżera kolejek jako ustawienie parametru MsgType na MQMT_REQUEST.
- Jeśli ani MQRO_PAN, ani MQRO_NAN nie są ustawione, odpowiedź nie jest wysyłana.

Formatowanie

Ustaw wartość MQFMT_RF_HEADER_2

MsgId

To pole jest zwykle ustawiane na wartość MQMI_NONE, dzięki czemu menedżer kolejek generuje unikalną wartość.

CorrelId

To pole może być ustawione na dowolną wartość. Jeśli tożsamość nadawcy zawiera element CorrelId, należy określić tę wartość wraz z atrybutem MQRO_PASS_CORREL_ID w polu Raport , aby upewnić się, że jest ona ustawiona we wszystkich komunikatach odpowiedzi wysyłanych przez menedżer kolejek do nadawcy.

Kolejka_zwrotna

To pole definiuje kolejkę, do której mają być wysyłane odpowiedzi (jeśli istnieją). Może to być kolejka nadawcy. Ma to tę zaletę, że parametr QName może zostać pominięty w komunikacie. Jeśli jednak odpowiedzi mają zostać wysłane do innej kolejki, wymagany jest parametr QName .

ReplyToQMgr

To pole definiuje menedżer kolejek dla odpowiedzi. Jeśli to pole pozostanie puste (wartość domyślna), lokalny menedżer kolejek umieszcza własną nazwę w tym polu.

Ustawienia MQMD dla publikacji przekazywanych przez menedżera kolejek

Menedżer kolejek używa tych ustawień pól w deskrypcji komunikatu (MQMD), gdy wysyła on publikację do subskrybenta. Wszystkie pozostałe pola w strukturze MQMD są ustawiane na wartości domyślne.

Raport

Opcja Raport jest ustawiona na wartość MQRO_NONE.

MsgType

Parametr MsgType jest ustawiony na wartość MQMT_DATAGRAM.

Utrata ważności

Utrata ważności jest ustawiana na wartość w komunikacie Publish otrzymanego od publikatora. W przypadku zachowanego komunikatu czas pozostający do spłaty jest skrócony o przybliżony czas, przez jaki komunikat był w menedżerze kolejek.

Formatowanie

Format jest ustawiony na wartość MQFMT_RF_HEADER_2

MsgId

Wartość MsgId jest ustawiona na unikalną wartość.

CorrelId

Jeśli element CorrelId jest częścią tożsamości subskrybenta, jest to wartość określona przez subskrybenta podczas rejestrowania. W przeciwnym razie jest to wartość niezerowa wybrana przez menedżer kolejek.

Priorytet

Priorytet przyjmuje wartość ustawioną przez publikator lub jako rozwiązana, jeśli publikator określił wartość MQPRI_PRIORITY_AS_Q_DEF.

Trwałość

Trwałość przyjmuje wartość ustawioną przez publikator lub jako rozwiązana, jeśli publikator określił wartość MQPER_PERSISTENCE_AS_Q_DEF, o ile nie określono inaczej w komunikacie Register Subscriber dla subskrybenta, do którego ta publikacja jest wysyłana.

Kolejka_zwrotna

Wartość ReplyToQ jest pusta.

ReplyToQMgr

ReplyToMenedżer kolejek jest ustawiony na nazwę menedżera kolejek.

UserIdentifier

UserIdentifier to identyfikator użytkownika subskrybenta, który jest ustawiany podczas rejestrowania subskrybenta.

AccountingToken

AccountingToken to znacznik rozliczania subskrybenta, który jest ustawiany po raz pierwszy zarejestrowany subskrybent.

Dane_tożsamości_aplikacji

Dane aplikacjiApplIdentity są danymi tożsamości aplikacji subskrybenta, które są ustawiane po raz pierwszy zarejestrowanej subskrybentowi.

PutApplType,

PutApplTyp jest ustawiony na wartość MQAT_BROKER.

PutApplName,

PutApplNazwa jest ustawiona na pierwsze 28 znaków nazwy menedżera kolejek.

PutDate

PutDate to data umieszczenia komunikatu.

PutTime

PutTime to czas, w którym komunikat został umieszczony.

ApplOriginData.

Parametr ApplOriginData jest ustawiony na wartości puste.

Ustawienia MQMD w komunikatach odpowiedzi menedżera kolejek

Menedżer kolejek używa tych ustawień pól w deskrypcji komunikatu (MQMD) podczas wysyłania odpowiedzi na komunikat z publikacją. Wszystkie pozostałe pola w strukturze MQMD są ustawiane na wartości domyślne.

Raport

Raport jest ustawiony na wszystkie zera.

MsgType

Parametr MsgType jest ustawiony na wartość MQMT_REPLY.

Formatowanie

Format jest ustawiony na wartość MQFMT_RF_HEADER_2

MsgId

Ustawienie wartości MsgId zależy od opcji Report w oryginalnym komunikacie komendy. Domyślnie jest ona ustawiona na wartość MQMI_NONE, dzięki czemu menedżer kolejek generuje unikalną wartość.

CorrelId

Ustawienie parametru CorrelId zależy od opcji Report w oryginalnym komunikacie komendy. Domyślnie oznacza to, że parametr CorrelId jest ustawiony na taką samą wartość, jak wartość MsgId komunikatu komendy. Można go używać do korelowania komend z ich odpowiedziami.

Priorytet

Wartość Priorytet jest ustawiona na tę samą wartość, co w oryginalnym komunikacie komendy.

Trwałość

Trwałość jest ustawiana na wartość ustawioną w oryginalnym komunikacie komendy.

Utrata ważności

Utrata ważności jest ustawiona na tę samą wartość, co w oryginalnym komunikacie komendy odebrany przez menedżer kolejek.

PutApplType,

PutApplTyp jest ustawiony na wartość MQAT_BROKER.

PutApplName,

PutApplNazwa jest ustawiona na pierwsze 28 znaków nazwy menedżera kolejek.

Pozostałe pola kontekstu są ustawiane w taki sposób, jakby były generowane z opcją MQPMO_PASS_IDENTITY_CONTEXT.

Kodowanie komputera

W tej sekcji opisano strukturę pola *Encoding* w deskrypcorze komunikatu.

Podsumowanie pól w strukturze znajduje się w sekcji [“MQMD-deskryptor komunikatu”](#) na stronie 417 .

Pole *Encoding* jest 32-bitową liczbą całkowitą, która jest podzielona na cztery oddzielne podpola. Te podpola identyfikują:

- Kodowanie używane dla binarnych liczb całkowitych
- Kodowanie używane dla upakowanych liczb całkowitych
- Kodowanie używane dla liczb zmiennopozycyjnych
- Zarezerwowane bity

Każde podpole jest identyfikowane przez maskę bitową, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Zdefiniowane są następujące maski:

MQENC_INTEGER_MASK

Maska dla kodowania binarnego-liczba całkowita.

To podpole zajmuje pozycje od 28 do 31 w obrębie pola *Encoding* .

MQENC_DECIMAL_MASK

Maska dla kodowania spakowanego-dziesiętnego.

To podpole zajmuje pozycje od 24 do 27 w obrębie pola *Encoding* .

MQENC_FLOAT_MASK

Maska dla kodowania zmiennopozycyjnego.

To podpole zajmuje pozycje od 20 do 23 w obrębie pola *Encoding* .

Maska MQENC_RESERVED_MASK

Maska dla bitów zarezerwowanych.

To podpole zajmuje pozycje od 0 do 19 w obrębie pola *Encoding* .

Kodowanie binarno-całkowite

Następujące wartości są poprawne dla kodowania binarnego-liczba całkowita:

MQENC_INTEGER_UNDEFINED

Binarne liczby całkowite są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

MQENC_INTEGER_NORMAL

Binarne liczby całkowite są reprezentowane w tradycyjny sposób:

- Najmniej znaczący bajt w tym numerze ma najwyższy adres dowolnego z bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres
- Najmniej znaczący bit w każdym bajcie sąsiaduje z bajtem o kolejnym wyższym adresie; najbardziej znaczący bit w każdym bajcie sąsiaduje z bajtem z następnym dolnym adresem

MQENC_INTEGER_REVERSED

Binarne liczby całkowite są reprezentowane w taki sam sposób, jak MQENC_INTEGER_NORMAL, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak w przypadku MQENC_INTEGER_NORMAL.

Packed-decimal-kodowanie całkowite

Następujące wartości są poprawne dla kodowania spakowanego-dziesiętnego-integer:

MQENC_DECIMAL_UNDEFINED

Upakowane dziesiętne liczby całkowite są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

MQENC_DECIMAL_NORMAL

Spakowane-dziesiętne liczby całkowite są reprezentowane w tradycyjny sposób:

- Każda cyfra dziesiętna w postaci drukowalnej liczby jest reprezentowana w upakowanym formacie dziesiętnym przez pojedynczą cyfrę szesnastkową z zakresu od X' 0 'do X' 9'. Każda cyfra szesnastkowa zajmuje cztery bity, a więc każdy bajt w upakowanej liczbie dziesiętnej reprezentuje dwie cyfry dziesiętne w postaci drukowalnej liczby.
- Najmniej znaczący bajt w upakowanej liczbie dziesiętnej to bajt, który zawiera najmniej znaczącą cyfrę dziesiętną. W tym bajcie, najbardziej znaczące cztery bity zawierają najmniej znaczącą cyfrę dziesiętną, a najmniej znaczące cztery bity zawierają znak. Znakiem jest X'C '(dodatni), X'D' (ujemny) lub X'F' (niepodpisany).
- Najmniej znaczący bajt w tym numerze ma najwyższy adres dowolnego z bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie sąsiaduje z bajtem o kolejnym wyższym adresie; najbardziej znaczący bit w każdym bajcie sąsiaduje z bajtem z następnym dolnym adresem.

MQENC_DECIMAL_REVERSED

Spakowane liczby całkowite są reprezentowane w taki sam sposób jak MQENC_DECIMAL_NORMAL, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak MQENC_DECIMAL_NORMAL.

Kodowanie zmiennopozycyjne

Następujące wartości są poprawne dla kodowania zmiennopozycyjnego:

MQENC_FLOAT_UNDEFINED

Liczby zmiennopozycyjne są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

MQENC_FLOAT_IEEE_NORMAL

Liczby zmiennopozycyjne są reprezentowane przy użyciu standardu IEEE³format zmiennopozycyjny, z liczbą bajtów ustawionych w następujący sposób:

- Najmniej znaczący bajt w mantysie ma najwyższy adres dowolnego z bajtów w liczbie; bajt zawierający wykładnik ma najniższy adres
- Najmniej znaczący bit w każdym bajcie sąsiaduje z bajtem o kolejnym wyższym adresie; najbardziej znaczący bit w każdym bajcie sąsiaduje z bajtem z następnym dolnym adresem

Szczegółowe informacje na temat kodowania zmiennopozycyjnego IEEE można znaleźć w standardzie IEEE 754.

MQENC_FLOAT_IEEE_REVERSED

Liczby zmiennopozycyjne są reprezentowane w taki sam sposób, jak MQENC_FLOAT_IEEE_NORMAL, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390

Liczby zmiennopozycyjne są reprezentowane przy użyciu standardowego formatu zmiennopozycyjnego System/390 . Jest on używany również przez system System/370.

Konstruowanie kodowania

Aby skonstruować wartość dla pola *Encoding* w strukturze MQMD, odpowiednie stałe opisujące wymagane kodowania mogą zostać dodane razem (nie należy dodawać tej samej stałej więcej niż raz) lub łączyć je za pomocą operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

³ Instytut Inżynierii Elektrycznej i Elektroniki

Niezależnie od tego, która metoda jest używana, należy połączyć tylko jeden z kodowań MQENC_INTEGER_* z jednym z kodowań MQENC_DECIMAL_* i jednym z kodowań MQENC_FLOAT_*.

Analizowanie kodowania

Pole *Encoding* zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić liczbę całkowitą, upakowaną liczbę dziesiętną lub kodowanie zmiennopozycyjne, muszą korzystać z jednej z opisanych technik.

Korzystanie z operacji bitowych

Jeśli język programowania obsługuje operacje bitowe, wykonaj następujące kroki:

1. Należy wybrać jedną z następujących wartości, zgodnie z wymaganym typem kodowania:
 - MQENC_INTEGER_MASK dla binarnego kodowania liczb całkowitych
 - MQENC_DECIMAL_MASK dla spakowanego kodowania liczb całkowitych
 - MQENC_FLOAT_MASK dla kodowania zmiennopozycyjnegoWywołaj wartość A.
2. Połącz pole *Encoding* z A za pomocą operacji bitowych AND (bitwise AND); wywołaj wynik B.
3. B jest wymaganym kodowaniem i może zostać przetestowany pod kątem równości z każdą z wartości, które są poprawne dla danego typu kodowania.

Korzystanie z arytmetyki

Jeśli język programowania *nie obsługuje* operacji bitowych, wykonaj następujące kroki, używając arytmetyki liczb całkowitych:

1. Należy wybrać jedną z następujących wartości, zgodnie z wymaganym typem kodowania:
 - 1 dla binarnego kodowania liczb całkowitych
 - 16 dla upakowanego dziesiętnego kodowania liczb całkowitych
 - 256 dla kodowania zmiennopozycyjnegoWywołaj wartość A.
2. Podziel wartość pola *Encoding* przez A ; Wywołaj wynik B.
3. Podziel B przez 16; wywołaj wynik C.
4. Pomnóż C przez 16 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. E jest wymaganym kodowaniem i może zostać przetestowany pod kątem równości z każdą z wartości, które są poprawne dla danego typu kodowania.

Podsumowanie kodowań architektury maszyn

Kodowania dla architektur maszyn są wyświetlane w programie [Tabela 117 na stronie 914](#).

Tabela 117. Podsumowanie kodowań dla architektur maszyn			
Architektura maszyny	Kodowanie binarnego liczb całkowitych	Pakowane-dziesiętne kodowanie całkowite	Kodowanie zmiennopozycyjne
IBM i	normalny	normalny	Normalne IEEE
Intel x86	Odwrotne	Odwrotne	IEEE odwrócone
PowerPC	normalny	normalny	Normalne IEEE

Tabela 117. Podsumowanie kodowań dla architektur maszyn (kontynuacja)			
Architektura maszyny	Kodowanie binarnego liczb całkowitych	Pakowane-dziesiętne kodowanie całkowite	Kodowanie zmiennopozycyjne
System/390	normalny	normalny	System/390

Opcje raportów i flagi komunikatów

W tej sekcji opisano pola *Report* i *MsgFlags*, które są częścią deskryptora komunikatu MQMD określonego w wywołaniach MQGET, MQPUT i MQPUT1.

Tematy w tej sekcji opisują:

- Struktura pola raportu i sposób jego przetwarzania przez menedżer kolejek
- Sposób analizowania pola raportu przez aplikację
- Struktura pola komunikatu-flagi

Więcej informacji na temat deskryptora komunikatu MQMD zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 417.

Struktura pola raportu

W tej sekcji opisano strukturę pola raportu.

Pole *Report* jest 32-bitową liczbą całkowitą, która jest podzielona na trzy oddzielne podpola. Te podpola identyfikują:

- Opcje raportu, które są odrzucane, jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Opcje raportów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Opcje raportu, które są akceptowane tylko wtedy, gdy spełnione są określone inne warunki

Każde podpole jest identyfikowane przez maskę bitową, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Bity w podpolu niekoniecznie przylegają do siebie. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Następujące maski są zdefiniowane w celu identyfikacji podpól:

MQRO_REJECT_UNSUP_MASK

Ta maska identyfikuje pozycje bitowe w polu *Report*, w którym opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, powodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_REPORT_OPTIONS_ERROR.

To podpole zajmuje pozycje bitowe 3, a 11 do 13.

MQRO_ACCEPT_UNSUP_MASK

Ta maska identyfikuje pozycje bitowe w polu *Report*, w których opcje raportów, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1. W tym przypadku zwrócony został kod zakończenia MQCC_WARNING z kodem przyczyny MQRC_UNKNOWN_REPORT_OPTION.

To podpole zajmuje pozycje bity od 0 do 2, od 4 do 10 i od 24 do 31.

W tym podpolu znajdują się następujące opcje raportu:

- MQRO_ACTIVITY,
- MQRO_COPY_MSG_ID_TO_CORREL_ID (Identyfikator 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

Ta maska identyfikuje pozycje bitowe w polu *Report*, w których opcje raportów, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem*, że spełnione są oba poniższe warunki:

- Komunikat jest przeznaczony dla menedżera kolejek zdalnych.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (oznacza to, że kolejka identyfikowana przez pola *ObjectQMgrName* i *ObjectName* w deskrytorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest kolejką lokalną transmisji).

Kod zakończenia MQCC_WARNING z kodem przyczyny MQRC_UNKNOWN_REPORT_OPTION są zwracane, jeśli te warunki są spełnione, a wartość MQCC_FAILED z kodem przyczyny MQRC_REPORT_OPTIONS_ERROR (jeśli nie).

To podpole zajmuje pozycje bity od 14 do 23.

W tym podpolu znajdują się następujące opcje raportu:

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Jeśli w polu *Report* określone są opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza poszczególne podpole z kolei za pomocą operacji bitowych AND, aby połączyć pole *Report* z maską dla tego podpole. Jeśli wynik tej operacji nie jest zerowy, zwracane są kody zakończenia i kody przyczyny opisane wcześniej.

Jeśli zostanie zwrócona wartość MQCC_WARNING, to nie jest ona zdefiniowana, który kod przyczyny jest zwracany, jeśli istnieją inne warunki ostrzeżenia.

Możliwość określania i akceptowanych opcji raportu, które nie są rozpoznawane przez lokalny menedżer kolejek, jest użyteczna podczas wysyłania komunikatu z opcją raportu rozpoznawaną i przetwarzaną przez *zdalny* menedżer kolejek.

Analizowanie pola raportu

Pole *Report* zawiera podpole. Z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu zażądał konkretnego raportu, musi użyć jednej z opisanych technik.

Korzystanie z operacji bitowych

Jeśli język programowania obsługuje operacje bitowe, wykonaj następujące kroki:

1. Wybierz jedną z następujących wartości, zgodnie z typem raportu, który ma zostać sprawdzony:
 - MQRO_COA_WITH_FULL_DATA dla raportu COA

- MQRO_COD_WITH_FULL_DATA dla raportu COD
- Raport MQRO_EXCEPTION_WITH_FULL_DATA dla wyjątku
- MQRO_EXPIRATION_WITH_FULL_DATA-raport o utracie ważności

Wywołaj wartość A.

W systemie z/OS należy użyć wartości MQRO_ * _WITH_DATA zamiast wartości MQRO_ * _WITH_FULL_DATA.

2. Połącz pole *Report* z A za pomocą operacji bitowych AND (bitwise AND); wywołaj wynik B.
3. Przetestuj B , aby uzyskać równość z każdą wartością, która jest możliwa dla tego typu raportu.

Na przykład, jeśli A to MQRO_EXCEPTION_WITH_FULL_DATA, testuj B na równość z każdym z poniższych, aby określić, co zostało określone przez nadawcę komunikatu:

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

Testy mogą być wykonywane w dowolnej kolejności, co jest najbardziej wygodne dla logiki aplikacji.

Użyj podobnej metody do przetestowania dla opcji MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID; wybierz wartość A , która z tych dwóch stałych jest odpowiednia, a następnie kontynuuj zgodnie z opisem.

Korzystanie z arytmetyki

Jeśli język programowania *nie obsługuje* operacji bitowych, wykonaj następujące kroki, używając arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, zgodnie z typem raportu, który ma zostać sprawdzony:
 - Raport MQRO_COA dla COA
 - Raport MQRO_COD dla COD
 - Raport MQRO_EXCEPTION dla wyjątku
 - MQRO_EXPIRATION-raport o utracie ważności

Wywołaj wartość A.

2. Divide the *Report* field by A ; call the result B.
3. Podziel B przez 8 ; Wywołaj wynik C.
4. Pomnóż C przez 8 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. Przetestuj E , aby uzyskać równość z każdą wartością, która jest możliwa dla tego typu raportu.

Na przykład, jeśli A to MQRO_EXCEPTION, testuj E na potrzeby równości z każdym z następujących, aby określić, co zostało określone przez nadawcę komunikatu:

- MQRO_NONE
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA

Testy mogą być wykonywane w dowolnej kolejności, co jest najbardziej wygodne dla logiki aplikacji.

Poniższy pseudocode ilustruje tę technikę dla komunikatów o wyjątkach:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
```

D = B - C*8
E = D*A

Użyj podobnej metody, aby przetestować opcje MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID. Wybierz wartość A, która z tych dwóch stałych jest odpowiednia, a następnie kontynuuj zgodnie z opisem poprzednio, ale zastępując wartość 8 w poprzednich krokach wartością 2.

Struktura pola komunikatu-flagi

W tej sekcji opisano strukturę pola flag komunikatu.

Pole *MsgFlags* jest 32-bitową liczbą całkowitą, która jest podzielona na trzy oddzielne podpola. Te podpola identyfikują:

- Flagi komunikatów, które są odrzucane, jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Flagi komunikatów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Flagi komunikatów, które są akceptowane tylko wtedy, gdy spełnione są pewne inne warunki

Uwaga: Wszystkie podpola w programie *MsgFlags* są zarezerwowane do użycia przez menedżer kolejek.

Każde podpole jest identyfikowane przez maskę bitową, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Następujące maski są zdefiniowane w celu identyfikacji podpól:

MQMF_REJECT_UNSUP_MASK

Ta maska identyfikuje pozycje bitowe w polu *MsgFlags*, w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, powodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_MSG_FLAGS_ERROR.

To podpole zajmuje pozycje bity od 20 do 31.

W tym podpolu znajdują się następujące opcje komunikatów:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBITED

MQMF_ACCEPT_UNSUP_MASK

Ta maska identyfikuje pozycje bitowe w polu *MsgFlags*, w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1. Kod zakończenia ma wartość MQCC_OK.

To podpole zajmuje pozycje bitowe od 0 do 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK

Ta maska identyfikuje pozycje bitowe w polu *MsgFlags*, w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem*, że spełnione są oba poniższe warunki:

- Komunikat jest przeznaczony dla menedżera kolejek zdalnych.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (oznacza to, że kolejka identyfikowana przez pola *ObjectQMGrName* i *ObjectName* w deskrytorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest kolejką lokalną transmisji).

Kod zakończenia MQCC_OK jest zwracany, jeśli te warunki są spełnione, a MQCC_FAILED z kodem przyczyny MQRC_MSG_FLAGS_ERROR (jeśli nie).

To podpole zajmuje pozycje bitowe od 12 do 19.

Jeśli w polu *MsgFlags* są określone opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza poszczególne podpola z kolei za pomocą operacji bitowych AND, aby połączyć pole *MsgFlags* z maską dla tego podpola. Jeśli wynik tej operacji nie jest zerowy, zwracane są kody zakończenia i kody przyczyny opisane wcześniej.

Konwersja danych

Ta kolekcja tematów opisuje interfejs do wyjścia konwersji danych oraz przetwarzanie wykonywane przez menedżer kolejek w przypadku, gdy wymagana jest konwersja danych.

Więcej informacji na temat konwersji danych znajduje się w dokumencie *Konwersja danych w programie IBM MQ* pod adresem <https://www.ibm.com/support/docview.wss?uid=swg27005729>.

Wyjście konwersji danych jest wywoływane jako część procesu przetwarzania wywołania MQGET w celu przekształcenia danych komunikatu aplikacji w reprezentację wymaganą przez aplikację odbierającą. Konwersja danych komunikatu aplikacji jest opcjonalna. Wymaga ona określenia opcji MQGMO_CONVERT w wywołaniu MQGET.

Opisywane są następujące tematy:

- Przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję MQGMO_CONVERT; patrz [“Przetwarzanie konwersji”](#) na stronie 919.
- Konwencje przetwarzania używane przez menedżera kolejek podczas przetwarzania wbudowanego formatu. Konwencje te są zalecane także dla programów zewnętrznych. Patrz sekcja [“Konwencje przetwarzania”](#) na stronie 920.
- Specjalne uwagi dotyczące przekształcania komunikatów raportów; patrz [“Konwersja komunikatów raportu”](#) na stronie 925.
- Parametry przekazane do wyjścia konwersji danych (data-conversion exit); patrz [“MQ_DATA_CONV_EXIT-wyjście konwersji danych”](#) na stronie 937.
- Wywołanie, którego można użyć z wyjścia w celu przekształcenia danych znakowych między różnymi reprezentacjami; patrz [“MQXCNVC-Przekształć znaki”](#) na stronie 931.
- Parametr struktury danych, który jest specyficzny dla wyjścia; patrz [“MQDXP-Dane-parametr wyjścia konwersji danych”](#) na stronie 925.

Przetwarzanie konwersji

Te informacje opisują przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję MQGMO_CONVERT.

Menedżer kolejek wykonuje następujące działania, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT i istnieje komunikat, który ma zostać zwrócony do aplikacji:

1. Jeśli co najmniej jedna z następujących wartości jest prawdziwa, konwersja nie jest konieczna:
 - Dane komunikatu znajdują się już w zestawie znaków i kodowaniu wymaganym przez aplikację, która wywołała wywołanie MQGET. Przed wywołaniem wywołania aplikacja musi ustawić pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* wywołania MQGET na wartości wymagane.
 - Długość danych komunikatu wynosi zero.
 - Długość parametru *Buffer* wywołania MQGET wynosi zero.

W takich przypadkach komunikat jest zwracany bez konwersji do aplikacji wywołujących wywołanie MQGET; wartości *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* są ustawiane na wartości w informacjach sterujących w komunikacie, a wywołanie kończy się jedną z następujących kombinacji kodu zakończenia i kodu przyczyny:

Kod zakończenia

MQCC_OK

MQCC_WARNING,

Kod przyczyny

MQRC_NONE

MQRC_TRUNCATED_MSG_ACCEPTED

Kod zakończenia

MQCC_WARNING,

Kod przyczyny

Funkcja MQRC_TRUNCATED_MSG_FAILED

Następujące kroki są wykonywane tylko wtedy, gdy zestaw znaków lub kodowanie danych komunikatu różni się od odpowiedniej wartości w parametrze *MsgDesc* i istnieją dane do przekształcenia:

2. Jeśli pole *Format* w informacjach sterujących w komunikacie ma wartość MQFMT_NONE, to komunikat jest zwracany bez konwersji, o kodzie zakończenia MQCC_WARNING i kodzie przyczyny MQRC_FORMAT_ERROR.

We wszystkich innych przypadkach przetwarzanie konwersji jest kontynuowane.

3. Komunikat zostanie usunięty z kolejki i umieszczony w tymczasowym buforze, który ma taką samą wielkość jak parametr *Buffer*. W przypadku operacji przeglądania komunikat jest kopiowany do tymczasowego buforu, a nie do usunięcia z kolejki.
4. Jeśli komunikat ma zostać obcięty, aby zmieścić się w buforze, wykonaj następujące czynności:
 - Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG *nie* została określona, zwracany jest komunikat bez konwersji z kodem zakończenia MQCC_WARNING, a kod przyczyny MQRC_TRUNCATED_MSG_FAILED.
 - Jeśli podano opcję MQGMO_ACCEPT_TRUNCATED_MSG *było*, kod zakończenia jest ustawiony na wartość MQCC_WARNING, kod przyczyny jest ustawiony na wartość MQRC_TRUNCATED_MSG_ACCEPTED, a przetwarzanie konwersji będzie kontynuowane.
5. Jeśli komunikat może być zakwaterowany w buforze bez obciążenia lub podano opcję MQGMO_ACCEPT_TRUNCATED_MSG, wykonaj następujące czynności:
 - Jeśli format jest formatem wbudowanym, bufor jest przekazywany do usługi konwersji danych menedżera kolejek.
 - Jeśli format nie jest formatem wbudowanym, bufor jest przekazywany do wyjścia napisanego przez użytkownika o tej samej nazwie, co format. Jeśli nie można znaleźć wyjścia, zwracany jest komunikat bez konwersji, o kodzie zakończenia MQCC_WARNING i kodzie przyczyny MQRC_FORMAT_ERROR.

Jeśli nie wystąpi błąd, dane wyjściowe z usługi konwersji danych lub z wyjścia napisanego przez użytkownika są komunikatem przekształconym, a kod zakończenia i kod przyczyny są zwracane do aplikacji wywołujących wywołanie MQGET.

6. Jeśli konwersja zakończy się pomyślnie, menedżer kolejek zwraca przekształcony komunikat do aplikacji. W tym przypadku kod zakończenia i kod przyczyny zwracane przez wywołanie MQGET są jedną z następujących kombinacji:

Kod zakończenia

MQCC_OK

MQCC_WARNING,

Kod przyczyny

MQRC_NONE

MQRC_TRUNCATED_MSG_ACCEPTED

Jeśli jednak konwersja jest wykonywana przy użyciu wyjścia napisanego przez użytkownika, mogą zostać zwrócone inne kody przyczyny, nawet jeśli konwersja jest pomyślna.

Jeśli konwersja nie powiedzie się, menedżer kolejek zwróci do aplikacji nieprzekształcony komunikat z polami *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* ustawionym na wartości w informacjach sterujących w komunikacie oraz z kodem zakończenia MQCC_WARNING.

Konwencje przetwarzania

Podczas przekształcania wbudowanego formatu menedżer kolejek postępuje zgodnie z opisanymi konwencjami przetwarzania.

Procedury zewnętrzne napisane przez użytkownika powinny również być zgodne z tymi konwencjami, chociaż nie jest to wymuszane przez menedżer kolejek. Wbudowane formaty przekształcone przez menedżera kolejek to:

- ADMINISTRATOR MQFMT_ADMIN
- MQFMT_CICS (tylko z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT, wersja 1
- MQFMT_EVENT, wersja 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 (tylko z/OS)
- MQFMT_XMIT_Q_HEADER

1. Jeśli komunikat zostanie rozwinięty podczas konwersji, a jego wielkość przekracza wartość parametru *Buffer* , zostanie wykonane następujące czynności:

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG *nie* została określona, zwracany jest komunikat bez konwersji, o kodzie zakończenia MQCC_WARNING i kodzie przyczyny MQRC_CONVERTED_MSG_TOO_BIG.
- Jeśli określono opcję MQGMO_ACCEPT_TRUNCATED_MSG *było* , komunikat jest obcinany, kod zakończenia jest ustawiany na wartość MQCC_WARNING, kod przyczyny jest ustawiony na wartość MQRC_TRUNCATED_MSG_ACCEPTED, a przetwarzanie konwersji będzie kontynuowane.

2. Jeśli dojdzie do obcięcia (przed lub podczas konwersji), liczba poprawnych bajtów zwracanych w parametrze *Buffer* może być *mniejsza niż* długość buforu.

Może się to zdarzyć na przykład wtedy, gdy 4-bajtowa liczba całkowita lub znak DBCS jest znakiem końca buforu. Niekompletny element informacji nie jest przekształcany, a te bajty w zwróconym komunikacie nie zawierają poprawnych informacji. Może to również wystąpić, jeśli komunikat, który został obcięty przed konwersją, został obcięty podczas konwersji.

Jeśli liczba zwróconych poprawnych bajtów jest mniejsza niż długość buforu, nieużywane bajty na końcu buforu są ustawione na wartości NULL.

3. Jeśli tablica lub łańcuch znaków jest końcem buforu, to konwertowane jest tyle danych, ile jest to możliwe; tylko określony element tablicy lub znak DBCS, który jest niekompletny, nie jest przekształcany; poprzedzające je elementy tablicy lub znaki są przekształcane.

4. Jeśli wystąpi obcięcie (przed lub podczas konwersji), długość zwrócona dla parametru *DataLength* jest długością komunikatu *unconverted* przed obcięciem.

5. Gdy łańcuchy są konwertowane między jednobajtowymi zestawami znaków (SBCS), dwubajtowymi zestawami znaków (DBCS) lub wielobajtowymi zestawami znaków (MBCS), łańcuchy mogą się rozszerzać lub zawierać kontrakt.

- W formatach PCF w formatach MQFMT_ADMIN, MQFMT_EVENT i MQFMT_PCF łańcuchy w strukturach MQCFST i MQCFSL są rozszerzać lub kontraktem w razie potrzeby, aby pomieścić łańcuch po konwersji.

W przypadku struktury łańcuchowej MQCFSL łańcuchy znajdujące się na liście mogą rozszerzać się lub zawierać kontrakt o różne kwoty. W takim przypadku menedżer kolejek dopełnia krótsze łańcuchy zawierające odstępy, aby ich długość była taka sama jak najdłuższy łańcuch po konwersji.

- W formacie MQFMT_REF_MSG_HEADER: łańcuchy adresowane przez pola *SrcEnvOffset*, *SrcNameOffset*, *DestEnvOffset* i *DestNameOffset*, w zależności od potrzeb, w zależności od potrzeb, w celu dostosowania do łańcuchów po konwersji.
- W formacie MQFMT_RF_HEADER w polu *NameValueString* jest rozwijana lub w razie potrzeby kontrakty, które muszą być zgodne z parami nazwy/wartości po konwersji.
- W strukturach o stałych wielkościach pól menedżer kolejek zezwala na rozszerzanie lub kontrastowanie łańcuchów w swoich stałych polach, pod warunkiem, że nie zostaną utracone żadne istotne informacje. W tym zakresie końcowe odstępy i znaki występujące po pierwszym znaku null w polu są traktowane jako nieistotne.
 - Jeśli łańcuch zostanie rozwinięty, ale tylko nieistotne znaki muszą zostać usunięte, aby pomieścić przekształcony łańcuch w polu, konwersja powiedzie się, a wywołanie zakończy się z kodem MQCC_OK i kodem przyczyny MQRC_NONE (przy założeniu, że nie wystąpiły inne błędy).
 - Jeśli łańcuch zostanie rozwinięty, ale przekształcony łańcuch wymaga usunięcia znaczących znaków, aby zmieścić się w tym polu, komunikat zostanie zwrócony bez konwersji, a wywołanie zakończy się łańcuchem MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_STRING_TOO_BIG.

Uwaga: Kod przyczyny MQRC_CONVERTED_STRING_TOO_BIG powoduje, że w tym przypadku podano, czy określono opcję MQGMO_ACCEPT_TRUNCATED_MSG.

- W przypadku umów łańcuchowych menedżer kolejek dopełnia łańcuch odstęпами na długość pola.
6. W przypadku komunikatów składających się z co najmniej jednej struktury nagłówek MQ, po której następują dane użytkownika, może zostać przekształcona jedna lub większa liczba struktur nagłówek, a pozostała część komunikatu nie jest dostępna. Jednak (z dwoma wyjątkami) pola *CodedCharSetId* i *Encoding* w każdej strukturze nagłówek zawsze poprawnie wskazują zestaw znaków i kodowanie danych, które są zgodne ze strukturą nagłówek.

Istnieją dwa wyjątki: struktury MQCIH i MQIIH, w których wartości w polach *CodedCharSetId* i *Encoding* w tych strukturach nie są znaczące. W przypadku tych struktur dane, które są zgodne ze strukturą, znajdują się w tym samym zestawie znaków i kodowaniu, co sama struktura MQCIH lub MQIIH.

7. Jeśli pola *CodedCharSetId* lub *Encoding* w informacjach sterujących o pobieranej wiadomości lub w parametrze *MsgDesc* określają wartości, które są niezdefiniowane lub nie są obsługiwane, menedżer kolejek może zignorować błąd, jeśli niezdefiniowana lub nieobsługiwana wartość nie musi być używana do konwersji komunikatu.

Na przykład, jeśli pole *Encoding* w komunikacie określa nieobsługiwane kodowanie zmiennopozycyjne, ale komunikat zawiera tylko dane będące liczbami całkowitymi lub zawiera dane zmiennopozycyjne, które nie wymagają konwersji (ponieważ kodowanie źródłowe i docelowe jest identyczne), błąd może nie zostać rozpoznany.

Jeśli błąd zostanie zdiagnozowany, komunikat zostanie zwrócony bez konwersji z kodem zakończenia MQCC_WARNING i z jednym z kodów przyczyny MQRC_SOURCE_*_ERROR lub MQRC_TARGET_*_ERROR (odpowiednio); pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* są ustawione na wartości w informacjach sterujących w komunikacie.

Jeśli błąd nie zostanie wykryty, a konwersja zakończy się pomyślnie, wartości zwracane w polach *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* są wartościami określonymi przez aplikację wywołującą wywołanie MQGET.

8. We wszystkich przypadkach, jeśli komunikat jest zwracany do aplikacji bez konwersji, kod zakończenia jest ustawiany na wartość MQCC_WARNING, a pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* są ustawione na wartości odpowiednie dla danych, które nie zostały przekształcone. Jest to wykonywane również dla parametru MQFMT_NONE.

Parametr *Reason* jest ustawiony na kod wskazujący, dlaczego nie można było wykonać konwersji, chyba że komunikat musiał zostać obcięty. Kody przyczyny związane z obcięciem mają pierwszeństwo przed kodami przyczyny związanymi z konwersją. (Aby określić, czy obcięty komunikat został przekształcony, należy sprawdzić wartości zwracane w polach *CodedCharSetId* i *Encoding* w parametrze *MsgDesc*).

W przypadku zdiagnozowania błędu zwracany jest konkretny kod przyczyny lub ogólny kod przyczyny MQRC_NOT_CONVERTED. Zwrócony kod przyczyny zależy od możliwości diagnostycznych bazy usługi konwersji danych.

9. Jeśli kod zakończenia MQCC_WARNING zostanie zwrócony, a istotny jest więcej niż jeden kod przyczyny, kolejność wykonywania operacji jest następująca:
 - a. Następujące przyczyny mają pierwszeństwo przed wszystkimi innymi; tylko jeden z powodów w tej grupie może powstać:
 - MQRC_SIGNAL_REQUEST_ACCEPTED
 - MQRC_TRUNCATED_MSG_ACCEPTED
 - b. Kolejność wykonywania operacji w pozostałych kodach przyczyny nie jest zdefiniowana.

10. Po zakończeniu wywołania MQGET:

- Następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:
 - MQRC_NONE
- Następujące kody przyczyny wskazują, że komunikat *mógł* zostać pomyślnie przekształcony (należy sprawdzić pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* , aby dowiedzieć się):
 - MQRC_MSG_MARKED_BROWSE_CO_OP
 - MQRC_TRUNCATED_MSG_ACCEPTED
- Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Następujące przetwarzanie jest specyficzne dla formatów wbudowanych; nie ma zastosowania do formatów zdefiniowanych przez użytkownika:

11. Z wyjątkiem następujących formatów:

- ADMINISTRATOR MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- Zdarzenie MQFMT_EVENT
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING

Żaden z wbudowanych formatów nie może być konwertowany z lub do zestawów znaków, które nie mają znaków SBCS dla znaków, które są poprawne w nazwach kolejek. Jeśli podejmowana jest próba wykonania takiej konwersji, komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR, jeśli jest to właściwe.

Zestaw znaków Unicode UCS-2 jest przykładem zestawu znaków, który nie ma znaków SBCS dla znaków, które są poprawne w nazwach kolejek.

12. Jeśli dane komunikatu dla wbudowanego formatu są obcinane, pola w komunikacie zawierające długości łańcuchów lub liczby elementów lub struktur *nie* są dostosowywane w taki sposób, aby odzwierciedlały długość danych rzeczywiście zwróconych do aplikacji. Wartości zwracane dla takich pól w danych komunikatu są wartościami mającymi zastosowanie do komunikatu *przed obcięciem*.

Podczas przetwarzania komunikatów, takich jak obcięty komunikat MQFMT_ADMIN, upewnij się, że aplikacja nie próbuje uzyskać dostępu do danych znajdujących się poza końcem zwróconych danych.

13. Jeśli nazwa formatu to MQFMT_DEAD_LETTER_HEADER, dane komunikatu rozpoczynają się od struktury MQDLH, po której ewentualnie następuje zero lub większa liczba bajtów danych komunikatu aplikacji. Format, zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane za pomocą pól *Format*, *CodedCharSetId* i *Encoding* w strukturze MQDLH na początku komunikatu. Ze względu na to, że struktura MQDLH i dane komunikatu aplikacji mogą mieć różne zestawy znaków i kodowania, jeden, drugi lub obie struktury MQDLH i dane komunikatu aplikacji mogą wymagać konwersji.

Menedżer kolejek przekształca najpierw strukturę MQDLH w razie potrzeby. Jeśli konwersja zakończy się pomyślnie, lub struktura MQDLH nie wymaga konwersji, menedżer kolejek sprawdza pola *CodedCharSetId* i *Encoding* w strukturze MQDLH, aby sprawdzić, czy konwersja danych komunikatu aplikacji jest wymagana. Jeśli wymagana jest konwersja, menedżer kolejek wywołuje wyjście napisane przez użytkownika z nazwą nadaną przez pole *Format* w strukturze MQDLH lub wykonuje samą konwersję (jeśli *Format* jest nazwą wbudowanego formatu).

Jeśli wywołanie MQGET zwróci kod zakończenia MQCC_WARNING, a kod przyczyny jest jednym z tych, które wskazują, że konwersja nie powiodła się, zastosowanie ma jedna z następujących sytuacji:

- Nie można przekształcić struktury MQDLH. W tym przypadku dane komunikatu aplikacji nie zostaną przekształcone.
- Struktura MQDLH została przekształcona, ale dane komunikatu aplikacji nie zostały przekształcone.

Aplikacja może sprawdzić wartości zwrócone w polach *CodedCharSetId* i *Encoding* w parametrze *MsgDesc*, a także wartości w strukturze MQDLH, aby określić, które z tych parametrów mają zastosowanie poprzednio.

14. Jeśli nazwa formatu to MQFMT_XMIT_Q_HEADER, dane komunikatu zaczynają się od struktury MQXQH, po której ewentualnie następuje zero lub większa liczba bajtów dodatkowych danych. Te dodatkowe dane są zwykle danymi komunikatu aplikacji (może to być zerowa długość), ale może być również jeden lub więcej dalszych struktur nagłówka MQ, na początku dodatkowych danych.

Struktura MQXQH musi znajdować się w zestawie znaków i kodowaniu menedżera kolejek. Format, zestaw znaków i kodowanie danych zgodnie ze strukturą MQXQH są nadawane przez pola *Format*, *CodedCharSetId* i *Encoding* w strukturze MQMD, które zawierają w zmaterializowanych tabelach MQXQH. Dla każdej kolejnej struktury nagłówka MQ, pola *Format*, *CodedCharSetId* i *Encoding* w strukturze opisują dane, które są zgodne z tą strukturą. Dane te są albo inną strukturą nagłówka MQ, albo danymi komunikatu aplikacji.

Jeśli dla komunikatu MQFMT_XMIT_Q_HEADER zostanie określona opcja MQGMO_CONVERT, dane komunikatu aplikacji i niektóre struktury nagłówka produktu MQ są przekształcane, *ale dane w strukturze MQXQH nie są*. Z tego powodu w przypadku powrotu z wywołania MQGET:

- Wartości pól *Format*, *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* opisują dane w strukturze MQXQH, a *nie* dane komunikatu aplikacji; dlatego wartości *nie* są takie same jak wartości określone przez aplikację, która wywołała wywołanie MQGET.

Wynika to z tego, że aplikacja, która wielokrotnie pobiera komunikaty z kolejki transmisji z określoną opcją MQGMO_CONVERT, musi zresetować pola *CodedCharSetId* i *Encoding* w parametrze *MsgDesc* do wartości wymaganych dla danych komunikatu aplikacji przed każdym wywołaniem MQGET.

- Wartości pól *Format*, *CodedCharSetId* i *Encoding* w ostatniej strukturze nagłówka MQ zawierają opis danych komunikatu aplikacji. Jeśli nie istnieją inne struktury nagłówka MQ, dane komunikatu aplikacji są opisywane przez te pola w strukturze MQMD w strukturze MQXQH. Jeśli konwersja powiedzie się, wartości będą takie same jak wartości określone w parametrze *MsgDesc* przez aplikację, która wywołała wywołanie MQGET.

Jeśli komunikat jest komunikatem o rozdzielaniu, w strukturze MQXQH występuje struktura MQDH (wraz z tablicami rekordów MQOR i MQPMR), po której po kolei może następować zero lub więcej struktur nagłówka MQ, a także zero lub więcej bajtów danych komunikatu aplikacji. Podobnie jak struktura MQXQH, struktura MQDH musi znajdować się w zestawie znaków i kodowaniu

menedżera kolejek, a nie jest ona konwertowana w wywołaniu MQGET, nawet jeśli określono opcję MQGMO_CONVERT.

Przetwarzanie opisanych wcześniej struktur MQXQH i MQDH jest przeznaczone przede wszystkim do użycia przez agenty kanału komunikatów podczas pobierania komunikatów z kolejek transmisji.

Konwersja komunikatów raportu

W ogólnym przypadku komunikat raportu może zawierać różne ilości danych komunikatu aplikacji, zgodnie z opcjami raportu określonymi przez nadawcę oryginalnego komunikatu. Jednak raport aktywności może zawierać dane, ale bez opcji raportu wymieniaj * _WITH_DATA w stałej.

W szczególności komunikat raportu może zawierać:

1. Brak danych komunikatu aplikacji
2. Niektóre z danych komunikatu aplikacji z oryginalnego komunikatu

Dzieje się tak wtedy, gdy nadawca oryginalnego komunikatu określa MQRO_ * _WITH_DATA, a komunikat jest dłuższy niż 100 bajtów.

3. Wszystkie dane komunikatu aplikacji z oryginalnego komunikatu

Dzieje się tak wtedy, gdy nadawca oryginalnego komunikatu określa MQRO_ * _WITH_FULL_DATA lub określa MQRO_ * _WITH_DATA, a komunikat ma wartość 100 bajtów lub krótszy.

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, kopiuje on nazwę formatu z oryginalnego komunikatu do pola *Format* w informacjach sterujących w komunikacie raportu. Nazwa formatu w komunikacie raportu może więc oznaczać, że długość danych różni się od długości rzeczywiście obecnej w komunikacie raportu (obserwacje 1 i 2 poprzednio).

Jeśli opcja MQGMO_CONVERT jest określona podczas pobierania komunikatu raportu:

- Dla przypadku 1 poprzednio, wyjście konwersji danych nie jest wywoływane (ponieważ komunikat raportu nie zawiera danych).
- W przypadku przypadku 3 poprzednio nazwa formatu poprawnie implikuje długość danych komunikatu.
- Jednak w przypadku 2 poprzednio wywołano wyjście konwersji danych w celu przekształcenia komunikatu, który jest *krótszy*, niż długość wynikający z nazwy formatu.

Ponadto kod przyczyny przekazany do wyjścia ma zwykle wartość MQRC_NONE (oznacza to, że kod przyczyny nie wskazuje, że komunikat został obcięty). Dzieje się tak dlatego, że dane komunikatu zostały obcięte przez *nadawcę* komunikatu raportu, a nie przez menedżer kolejek odbiorcy w odpowiedzi na wywołanie MQGET.

Ze względu na te możliwości wyjście konwersji danych musi *nie* używać nazwy formatu do odliczenia długości przekazywanych do niej danych; zamiast tego wyjście musi sprawdzić długość podanych danych i być przygotowane do konwersji danych *mniej* niż długość implikowana przez nazwę formatu. Jeśli dane mogą zostać przekształcone pomyślnie, kod zakończenia MQCC_OK i kod przyczyny MQRC_NONE muszą zostać zwrócone przez wyjście. Długość danych komunikatu, które mają zostać przekształcone, jest przekazywana do wyjścia jako parametr *InBufferLength*.

Interfejs programistyczny wrażliwy na produkt

MQDXP-Dane-parametr wyjścia konwersji danych

Struktura MQDXP jest parametrem, który jest przekazywany przez menedżer kolejek do wyjścia konwersji danych po wywołaniu wyjścia w celu przekształcenia danych komunikatu w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat wyjścia konwersji danych można znaleźć w opisie wywołania MQ_DATA_CONV_EXIT.

Dane znakowe w produkcie MQDXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Dane te są nadawane przez atrybut menedżera kolejek produktu *CodedCharSetId*. Dane liczbowe w MQDXP znajdują się w rodzimym kodowaniu komputera. Dane te są podawane przez komendę MQENC_NATIVE.

Tylko pola *DataLength*, *CompCode*, *Reason* i *ExitResponse* w produkcie MQDXP mogą zostać zmienione przez wyjście. Zmiany w innych polach są ignorowane. Jednak pole *DataLength* nie może zostać zmienione, jeśli przekształcany komunikat jest to segment, który zawiera tylko część komunikatu logicznego.

Gdy sterowanie powraca do menedżera kolejek z wyjścia, menedżer kolejek sprawdza wartości zwrócone w MQDXP. Jeśli zwrócone wartości nie są poprawne, menedżer kolejek kontynuuje przetwarzanie, tak jakby to wyjście zwróciło wartość MQXDR_CONVERSION_FAILED w *ExitResponse* ; Jednak menedżer kolejek ignoruje wartości pól *CompCode* i *Reason* zwracanych przez wyjście w tym przypadku, a zamiast tych wartości te pola miały wartość *input* dla wyjścia. Następujące wartości w tabeli MQDXP powodują, że przetwarzanie to ma miejsce:

- Pole *ExitResponse* nie MQXDR_OK, a nie MQXDR_CONVERSION_FAILED
- Pole *CompCode* nie MQCC_OK, a nie MQCC_WARNING
- Pole *DataLength* mniejsze niż zero lub pole *DataLength* zmienione, gdy przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.

W poniższej tabeli podsumowano pola w strukturze.

Tabela 118. Pola w MQDXP		
Pole	Opis	Temat
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>AppOptions</i>	Opcje aplikacji	AppOptions
<i>Encoding</i>	Kodowanie numeryczne wymagane przez aplikację	Kodowanie
<i>CodedCharSetId</i>	Zestaw znaków wymagany przez aplikację	CodedCharSetId
<i>DataLength</i>	Długość (w bajtach) danych komunikatu	DataLength
<i>CompCode</i>	Kod zakończenia	CompCode
<i>Reason</i>	Kod przyczyny kwalifikujący <i>CompCode</i>	Powód
<i>ExitResponse</i>	Odpowiedź z wyjścia	ExitResponse
<i>Hconn</i>	Uchwyt połączenia	Hconn
<i>pEntryPoints</i>	Adres struktury MQIEP	pEntryPunkty

Pola

Struktura MQDXP zawiera następujące pola: pola są opisane w kolejności alfabetycznej.

AppOptions

Typ: MQLONG

To jest kopia pola *Options* struktury MQGMO określonej przez aplikację wywołującą wywołanie MQGET. Wyjście może być konieczne, aby sprawdzić, czy podano opcję MQGMO_ACCEPT_TRUNCATED_MSG.

To jest pole wejściowe do wyjścia.

CodedCharSetId

Typ: MQLONG

Jest to identyfikator kodowanego zestawu znaków zestawu znaków wymagany przez aplikację wywołującą wywołanie MQGET. Więcej informacji zawiera pole *CodedCharSetId* w strukturze MQMD. Jeśli aplikacja określa wartość specjalną MQCCSI_Q_MGR w wywołaniu MQGET, menedżer kolejek zmienia ten identyfikator na rzeczywisty identyfikator zestawu znaków zestawu znaków używanego przez menedżer kolejek przed wywołaniem wyjścia.

Jeśli konwersja zakończy się pomyślnie, wyjście musi skopiować to pole do pola *CodedCharSetId* w deskrypcorze komunikatu.

To jest pole wejściowe do wyjścia.

CompCode

Typ: MQLONG

Po wywołaniu wyjścia zawiera on kod zakończenia, który jest zwracany do aplikacji, która wywołała wywołanie MQGET, jeśli wyjście nie robi nic. Zawsze jest to MQCC_WARNING, ponieważ albo komunikat został obcięty, albo komunikat wymaga konwersji, a to jeszcze nie zostało zrobione.

Po wyjściu z wyjścia to pole zawiera kod zakończenia, który ma zostać zwrócony do aplikacji w parametrze *CompCode* wywołania MQGET. Poprawne są tylko wartości MQCC_OK i MQCC_WARNING. W opisie pola *Reason* można znaleźć sugestie dotyczące sposobu, w jaki wyjście może ustawić to pole na wyjściu.

Jest to pole wejściowe/wyjściowe do wyjścia.

DataLength

Typ: MQLONG

Po wywołaniu wyjścia w tym polu znajduje się oryginalna długość danych komunikatu aplikacji. Jeśli komunikat został obcięty, aby zmieścić się w buforze udostępnionym przez aplikację, wielkość komunikatu dostarczanego do wyjścia jest *mniejsza* niż wartość parametru *DataLength*. Wielkość komunikatu dostarczanego do wyjścia jest zawsze podawana przez parametr *InBufferLength* wyjścia, niezależnie od tego, które nastąpiło obcięcie.

Obcięcie jest wskazywanych przez pole *Reason*, które ma wartość MQRC_TRUNCATED_MSG_ACCEPTED w przypadku wejścia do wyjścia.

Większość konwersji nie musi zmieniać tej długości, ale wyjście może to zrobić w razie potrzeby; wartość ustawiona przez wyjście jest zwracana do aplikacji w parametrze *DataLength* wywołania MQGET. Jednak ta długość *nie może* zostać zmieniona, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego. Jest to spowodowane tym, że zmiana długości spowoduje, że przesunięcia późniejszych segmentów w komunikacie logicznym są niepoprawne.

Należy zwrócić uwagę, że jeśli wyjście chce zmienić długość danych, należy pamiętać, że menedżer kolejek już zdecydował, czy dane komunikatu wpisują się do buforu aplikacji, na podstawie długości danych *bez konwersji*. Ta decyzja określa, czy komunikat jest usuwany z kolejki (lub przeniesiono kursor przeglądania, dla żądania przeglądania) i nie ma wpływu na zmianę długości danych spowodowaną przez konwersję. Z tego powodu zaleca się, aby wyjścia konwersji nie powodował zmiany długości danych komunikatu aplikacji.

Jeśli konwersja znaków oznacza zmianę długości, łańcuch może zostać przekształcony w inny łańcuch o tej samej długości w bajtach, obcinanie odstępów końcowych lub dopełnianie odstępami w razie potrzeby.

Wyjście nie jest wywoływane, jeśli komunikat nie zawiera danych komunikatu aplikacji, dlatego wartość *DataLength* jest zawsze większa niż zero.

Jest to pole wejściowe/wyjściowe do wyjścia.

Encoding

Typ: MQLONG

Kodowanie numeryczne wymagane przez aplikację.

Jest to kodowanie liczbowe, które jest wymagane przez aplikację wywołującą wywołanie MQGET. Więcej szczegółów zawiera pole *Encoding* w strukturze MQMD.

Jeśli konwersja zakończy się pomyślnie, wyjście kopiuje to pole do pola *Encoding* w deskrypcorze komunikatu.

To jest pole wejściowe do wyjścia.

ExitOptions

Typ: MQLONG

Jest to pole zastrzeżone; jego wartością jest 0.

ExitResponse

Typ: MQLONG

Odpowiedź z wyjścia. Wartość ta jest ustawiana przez wyjście w celu wskazania powodzenia lub innej konwersji. Musi to być jeden z następujących elementów:

MQXDR_OK

Konwersja powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca następujące informacje do aplikacji, która wywołała wywołanie MQGET:

- Wartość pola *CompCode* na wyjściu z wyjścia
- Wartość pola *Reason* na wyjściu z wyjścia
- Wartość pola *DataLength* na wyjściu z wyjścia
- Zawartość buforu wyjściowego wyjścia *OutBuffer*. Liczba zwróconych bajtów jest mniejsza od parametru *OutBufferLength* wyjścia, a wartość pola *DataLength* na wyjściu z wyjścia.

Jeśli pola *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia mają wartość *both* bez zmian, menedżer kolejek zwraca:

- Wartość pól *Encoding* i *CodedCharSetId* w strukturze MQDXP w *danych wejściowych* do wyjścia.

Jeśli jeden lub oba pola *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia zostały zmienione, menedżer kolejek zwraca następujące dane:

- Wartość pól *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia na wyjściu wyjścia z wyjścia.

MQXDR_CONVERSION_FAILED

Konwersja nie powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca następujące informacje do aplikacji, która wywołała wywołanie MQGET:

- Wartość pola *CompCode* na wyjściu z wyjścia
- Wartość pola *Reason* na wyjściu z wyjścia
- Wartość pola *DataLength* w *danych wejściowych* do wyjścia
- Zawartość buforu wejściowego wyjścia *InBuffer*. Liczba zwróconych bajtów jest podawana przez parametr *InBufferLength*.

Jeśli wyjście zostało zmienione *InBuffer*, wyniki są niezdefiniowane.

ExitResponse to pole wyjściowe z wyjścia.

Hconn

Typ: MQHCONN

Jest to uchwyt połączenia, który może być używany w wywołaniu MQXCNCV. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

pEntryPoints

Typ: PMQIEP

Adres struktury MQIEP, za pośrednictwem której mogą być wykonywane wywołania MQI i DCI.

Reason

Typ: MQLONG

Kod przyczyny kwalifikujący *CompCode*.

Po wywołaniu wyjścia zawiera on kod przyczyny, który jest zwracany do aplikacji, która wydała wywołanie MQGET, jeśli program obsługi wyjścia nie zdecyduje się na nic. Wśród możliwych wartości można znaleźć wartość MQRC_TRUNCATED_MSG_ACCEPTED, która wskazuje, że komunikat został obcięty w celu dopasowania do buforu udostępnionego przez aplikację i MQRC_NOT_CONVERTED, co oznacza, że komunikat wymaga konwersji, ale nie zostało to jeszcze wykonane.

W przypadku wyjścia z wyjścia to pole zawiera przyczynę, która ma zostać zwrócona do aplikacji w parametrze *Reason* wywołania MQGET. Zalecane jest następujące ustawienie:

- Jeśli wartość parametru *Reason* ma wartość MQRC_TRUNCATED_MSG_ACCEPTED w przypadku wejścia do wyjścia, pola *Reason* i *CompCode* nie mogą być zmieniane, bez względu na to, czy konwersja powiodła się, czy też nie.

(Jeśli pole *CompCode* nie ma wartości MQCC_OK, aplikacja, która pobiera komunikat, może zidentyfikować błąd konwersji, porównując zwrócone wartości *Encoding* i *CodedCharSetId* w deskrytorze komunikatu z żądanymi wartościami; w przeciwieństwie do tego aplikacja nie może odróżnić obciętej wiadomości od komunikatu, który dopasował bufor. Z tego powodu wartość MQRC_TRUNCATED_MSG_ACCEPTED musi zostać zwrócona w preferencjach z przyczyn wskazujących na niepowodzenie konwersji.)
- Jeśli program *Reason* ma jakąkolwiek inną wartość na wejściu do wyjścia:
 - Jeśli konwersja powiedzie się, parametr *CompCode* musi być ustawiony na wartość MQCC_OK, a parametr *Reason* na wartość MQRC_NONE.
 - Jeśli konwersja nie powiedzie się lub komunikat zostanie rozwinięty i musi zostać obcięty w celu dopasowania do buforu, wartość *CompCode* musi zostać ustawiona na wartość MQCC_WARNING (lub pozostaw bez zmian), a parametr *Reason* na jedną z wymienionych wartości w celu wskazania natury niepowodzenia.

Należy pamiętać, że jeśli komunikat po konwersji jest zbyt duży dla buforu, musi zostać obcięty tylko wtedy, gdy aplikacja, która wywołała wywołanie MQGET, określiła opcję MQGMO_ACCEPT_TRUNCATED_MSG:
 - Jeśli ta opcja została określona, zwracana jest przyczyna MQRC_TRUNCATED_MSG_ACCEPTED.
 - Jeśli ta opcja nie została określona, komunikat zostanie zwrócony bez konwersji, a kod przyczyny MQRC_CONVERTED_MSG_TOO_BIG.

Wymienione kody przyczyny są zalecane do użycia przez wyjście w celu wskazania przyczyny niepowodzenia konwersji, ale wyjście może zwrócić inne wartości z zestawu kodów MQRC_*, jeśli jest to uznane za odpowiednie. Dodatkowo, zakres wartości MQRC_APPL_FIRST za pomocą MQRC_APPL_LAST jest przydzielany do użycia przez wyjście w celu wskazania warunków, które program obsługi wyjścia chce komunikować z aplikacją wywołującym wywołanie MQGET.

Uwaga: Jeśli komunikat nie może zostać pomyślnie przekształcony, wyjście *musi* zwracać wartość MQXDR_CONVERSION_FAILED w polu *ExitResponse*, aby menedżer kolejek mógł zwrócić nieprzekształcone komunikaty. Jest to prawda, niezależnie od kodu przyczyny zwróconego w polu *Reason*.

MQRC_APPL_FIRST

(900, X'384 ') Najniższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

MQRC_APPL_LAST

(999, X'3E7') Najwyższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

MQRC_NOT_CONVERTED

(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

MQRC_SOURCE_CCSD_ERROR, BŁĄD

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

MQRC_SOURCE_DECIMAL_ENC_ERROR

(2113, X'841 ') Zpakowane kodowanie dziesiętne w komunikacie nie zostało rozpoznane.

MQRC_SOURCE_FLOAT_ENC_ERROR, BŁĄD

(2114, X'842 ') Kodowanie zmiennopozycyjne w komunikacie nie zostało rozpoznane.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

MQRC_TARGET_CCSD_ERROR

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

MQRC_TARGET_DECIMAL_ENC_ERROR,

(2117, X'845 ') Zpakowane-kodowanie dziesiętne określone przez odbiornik nierozpoznany.

MQRC_TARGET_FLOAT_ENC_ERROR, BŁĄD

(2118, X'846 ') Kodowanie zmiennopozycyjne określone przez odbiornik nie jest rozpoznawane.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

MQRC_TRUNCATED_MSG_ACCEPTED

(2079, X'81F') Zwrócona została obcięta wiadomość (przetwarzanie zostało zakończone).

Jest to pole wejściowe/wyjściowe do wyjścia.

StrucId

Typ: MQCHAR4

Identyfikator struktury. Wartość musi być następująca:

MQDXP_STRUC_ID

Identyfikator struktury parametru wyjścia konwersji danych.

W przypadku języka programowania C zdefiniowana jest również stała MQDXP_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQDXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe do wyjścia.

Version

Typ: MQLONG

Numer wersji struktury. Wartość musi być następująca:

MQDXP_VERSION_1

Numer wersji struktury parametru wyjścia konwersji danych.

Następująca stała określa numer wersji bieżącej wersji:

MQDXP_CURRENT_VERSION

Bieżąca wersja struktury parametru wyjścia konwersji danych.

Uwaga: Gdy zostanie wprowadzona nowa wersja tej struktury, układ istniejącej części nie jest zmieniany. W związku z tym wyjście musi sprawdzić, czy pole *Version* jest równe lub większe od najniższej wersji, która zawiera pola, które mają być używane przez wyjście.

To jest pole wejściowe do wyjścia.

Deklaracja C

```
typedef struct tagMQDXP MQDXP;
```

```

struct tagMQDXP {
    MQCHAR4  StrucId;          /* Structure identifier */
    MQLONG   Version;          /* Structure version number */
    MQLONG   ExitOptions;      /* Reserved */
    MQLONG   AppOptions;       /* Application options */
    MQLONG   Encoding;         /* Numeric encoding required by
                               application */
    MQLONG   CodedCharSetId;    /* Character set required by application */
    MQLONG   DataLength;       /* Length in bytes of message data */
    MQLONG   CompCode;         /* Completion code */
    MQLONG   Reason;           /* Reason code qualifying CompCode */
    MQLONG   ExitResponse;      /* Response from exit */
    MQHCONN  Hconn;            /* Connection handle */
    PMQIEP   pEntryPoints;     /* Address of the MQIEP structure */
};

```

Deklaracja języka COBOL (tylko IBM i)

```

**  MQDXP structure
10  MQDXP.
**    Structure identifier
15  MQDXP-STRUCID      PIC X(4).
**    Structure version number
15  MQDXP-VERSION     PIC S9(9) BINARY.
**    Reserved
15  MQDXP-EXITOPTIONS  PIC S9(9) BINARY.
**    Application options
15  MQDXP-APPOPTIONS  PIC S9(9) BINARY.
**    Numeric encoding required by application
15  MQDXP-ENCODING     PIC S9(9) BINARY.
**    Character set required by application
15  MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**    Length in bytes of message data
15  MQDXP-DATALength  PIC S9(9) BINARY.
**    Completion code
15  MQDXP-COMPCODE     PIC S9(9) BINARY.
**    Reason code qualifying COMPCODE
15  MQDXP-REASON       PIC S9(9) BINARY.
**    Response from exit
15  MQDXP-EXITRESPONSE PIC S9(9) BINARY.
**    Connection handle
15  MQDXP-HCONN        PIC S9(9) BINARY.

```

Deklaracja asemblera System/390

```

MQDXP          DSECT
MQDXP_STRUCID  DS   CL4  Structure identifier
MQDXP_VERSION  DS   F    Structure version number
MQDXP_EXITOPTIONS DS   F    Reserved
MQDXP_APPOPTIONS DS   F    Application options
MQDXP_ENCODING DS   F    Numeric encoding required by application
MQDXP_CODEDCHARSETID DS   F    Character set required by application
MQDXP_DATALength DS   F    Length in bytes of message data
MQDXP_COMPCODE DS   F    Completion code
MQDXP_REASON   DS   F    Reason code qualifying COMPCODE
MQDXP_EXITRESPONSE DS   F    Response from exit
MQDXP_HCONN    DS   F    Connection handle
*
MQDXP_LENGTH   EQU   *-MQDXP
ORG   MQDXP
MQDXP_AREA     DS    CL(MQDXP_LENGTH)

```

MQXCNCV-Przekształć znaki

Wywołanie MQXCNCV konwertuje znaki z jednego zestawu znaków na inny przy użyciu języka programowania C.

To wywołanie jest częścią interfejsu DCI (Data Conversion Interface) produktu IBM MQ , który jest jednym z interfejsów środowiska produktu IBM MQ .

Uwaga: Wywołanie może być używane zarówno z aplikacji, jak i ze środowisk wyjścia konwersji danych.

Składnia

MQXCNCV (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parametry

Hconn

Typ: MQHCONN-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

W przypadku wyjścia konwersji danych program *Hconn* jest zwykle uchwyt przekazywany do wyjścia konwersji danych w polu *Hconn* struktury MQDXP. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

W systemie IBM imożna określić następującą wartość specjalną dla *Hconn*:

MQHC_DEF_HCONN

Domyślny uchwyt połączenia.

W przypadku uruchamiania aplikacji CICS TS 3.2 lub nowszej należy upewnić się, że program obsługi wyjścia konwersji znaków, który wywołuje wywołanie MQXCNCV, jest zdefiniowany jako OPENAPI. Ta definicja zapobiega wystąpieniu błędu MQRC_HCONN_ERROR 2018 wywołanego przez niepoprawne połączenie i umożliwia zakończenie operacji MQGET.

Opcje

Typ: MQLONG-wejście

Opcje, które sterują działaniem MQXCNCV.

Można podać zero lub więcej opcji opisanych w tej sekcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Domyślna-opcja konwersji: Poniższa opcja steruje użyciem domyślnej konwersji znaków:

MQDCC_DEFAULT_CONVERSION

Konwersja domyślna.

Ta opcja określa, że domyślna konwersja znaków może być używana, jeśli jeden lub oba zestawy znaków określone w wywołaniu nie są obsługiwane. Umożliwia to menedżerowi kolejek korzystanie z domyślnego zestawu znaków określonego przez instalację, który przybliża określony zestaw znaków podczas przekształcania łańcucha.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków w celu przekształcenia łańcucha jest niepoprawna konwersja niektórych znaków. Można tego uniknąć, używając w łańcuchu tylko znaków, które są wspólne zarówno dla określonego zestawu znaków, jak i domyślnego zestawu znaków.

Domyślne zestawy znaków są definiowane przy użyciu opcji konfiguracyjnej, gdy menedżer kolejek jest zainstalowany lub zrestartowany.

Jeśli wartość MQDCC_DEFAULT_CONVERSION nie jest określona, menedżer kolejek używa tylko określonych zestawów znaków w celu przekształcenia łańcucha, a wywołanie nie powiedzie się, jeśli jeden lub oba zestawy znaków nie są obsługiwane.

Ta opcja jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Opcja dopełnienia: Poniższa opcja umożliwia menedżerowi kolejek dopełnianie przekształconego łańcucha za pomocą odstępów lub odrzucania nieistotnych znaków końcowych, tak aby przekształcony łańcuch pasował do buforu docelowego:

MQDCC_FILL_TARGET_BUFFER

Bufor docelowy wypełnienia.

Ta opcja wymaga, aby konwersja była wypełniona w taki sposób, aby bufor docelowy został całkowicie wypełniony:

- Jeśli po przekształceniu kontrakty łańcuchowe są przekształcane, to w celu wypełnienia buforu docelowego dodawane są odstępy końcowe.
- Jeśli łańcuch zostanie rozwinięty po przekształceniu, znaki końcowe, które nie są znaczące, zostaną odrzucone, aby przekształcony łańcuch pasował do buforu docelowego. Jeśli ta operacja może zostać wykonana pomyślnie, wywołanie zakończy się z kodem MQCC_OK i kodem przyczyny MQRC_NONE.

Jeśli w buforze docelowym znajduje się zbyt mało znaczących znaków końcowych, to w buforze docelowym znajduje się wiele łańcuchów, które mogą zmieścić się w tym łańcuchu, a wywołanie kończy się łańcuchem MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG.

Nieistotne znaki to:

- Odstępy końcowe
- Znaki następujące po pierwszym znaku o kodzie zero w łańcuchu (ale z wyjątkiem pierwszego znaku o kodzie zero)
- Jeśli łańcuch, *TargetCCSID* i *TargetLength* są takie, że bufor docelowy nie może być całkowicie ustawiony z poprawnymi znakami, wywołanie kończy się niepowodzeniem z błędem MQCC_FAILED i kodem przyczyny MQRC_TARGET_LENGTH_ERROR. Może się tak zdarzyć, gdy *TargetCCSID* jest zestawem znaków DBCS (na przykład UCS-2, ale *TargetLength* określa długość, która jest nieparzystą liczbą bajtów.
- Wartość *TargetLength* może być mniejsza lub większa niż *SourceLength*. W przypadku powrotu z tabeli MQXCNCV, produkt *DataLength* ma taką samą wartość, jak *TargetLength*.

Jeśli ta opcja nie jest określona:

- W razie potrzeby łańcuch może zostać zamówiony lub rozwinięty w buforze docelowym. Nieistotne znaki końcowe nie są dodawane ani usuwane.

Jeśli przekształcony łańcuch mieści się w buforze docelowym, wywołanie kończy się łańcuchem MQCC_OK i kodem przyczyny MQRC_NONE.

Jeśli przekształcony łańcuch jest zbyt duży dla buforu docelowego, tak samo jak w buforze docelowym jest umieszczany łańcuch w postaci łańcucha, a wywołanie kończy się łańcuchem MQCC_WARNING, a kod przyczyny MQRC_CONVERTED_MSG_TOO_BIG. W tym przypadku można zwrócić uwagę na mniejszą liczbę bajtów niż *TargetLength*.

- Wartość *TargetLength* może być mniejsza lub większa niż *SourceLength*. W przypadku powrotu z MQXCNCV, *DataLength* jest mniejsze lub równe *TargetLength*.

Ta opcja jest obsługiwana w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Opcje kodowania: opisywane opcje mogą być używane do określenia kodowania liczb całkowitych dla łańcuchów źródłowych i docelowych. Odpowiednie kodowanie jest używane *tylko*, gdy odpowiedni identyfikator zestawu znaków wskazuje, że reprezentacja zestawu znaków w pamięci głównej jest zależna od kodowania używanego dla binarnych liczb całkowitych. Ma to wpływ tylko na niektóre wielobajtowe zestawy znaków (na przykład zestawy znaków UCS-2).

Kodowanie jest ignorowane, jeśli zestaw znaków to zestaw znaków jednobajtowych (SBCS) lub zestaw znaków wielobajtowych z reprezentacją w głównej pamięci masowej, która nie jest zależna od kodowania liczb całkowitych.

Należy podać tylko jedną z wartości MQDCC_SOURCE_*, w połączeniu z jedną z wartości MQDCC_TARGET_*:

MQDCC_SOURCE_ENC_NATIVE

Kodowanie źródłowe jest domyślne dla środowiska i języka programowania.

MQDCC_SOURCE_ENC_NORMAL

Kodowanie źródłowe jest normalne.

MQDCC_SOURCE_ENC_REVERSED

Kodowanie źródłowe zostało odwrócone.

MQDCC_SOURCE_ENC_UNDEFINED

Kodowanie źródłowe jest niezdefiniowane.

MQDCC_TARGET_ENC_NATIVE

Kodowanie docelowe jest wartością domyślną dla środowiska i języka programowania.

MQDCC_TARGET_ENC_NORMAL

Kodowanie docelowe jest normalne.

MQDCC_TARGET_ENC_REVERSED

Kodowanie docelowe jest odwrócone.

MQDCC_TARGET_ENC_UNDEFINED

Kodowanie docelowe jest niezdefiniowane.

Zdefiniowane wcześniej wartości kodowania można dodać bezpośrednio do pola *Options* . Jeśli jednak kodowanie źródłowe lub docelowe jest uzyskiwane z pola *Encoding* w strukturze MQMD lub innej strukturze, należy wykonać następujące przetwarzanie:

1. Kodowanie liczb całkowitych musi zostać wyodrębnione z pola *Encoding* , eliminując kodowanie zmiennopozycyjne i upakowane dziesiętne. Więcej informacji na temat tego sposobu można znaleźć w sekcji [“Analizowanie kodowania”](#) na stronie 914 .
2. Kodowanie całkowitoliczbowe wynikające z kroku 1 musi zostać pomnożone przez odpowiedni współczynnik zanim zostanie dodane do pola *Options* . Są to następujące czynniki:
 - MQDCC_SOURCE_ENC_FACTOR dla kodowania źródłowego
 - MQDCC_TARGET_ENC_FACTOR dla kodowania docelowego

Poniższy przykładowy kod ilustruje, w jaki sposób można go zakodować w języku programowania C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_SOURCE_ENC_FACTOR
        + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
          * MQDCC_TARGET_ENC_FACTOR;
```

Jeśli parametr nie zostanie określony, opcje kodowania są domyślnie niezdefiniowane (MQDCC_*_ENC_UNDEFINED). W większości przypadków nie ma to wpływu na pomyślne zakończenie wywołania MQXCNCV. Jeśli jednak odpowiedni zestaw znaków to zestaw znaków wielobajtowych z reprezentacją zależną od kodowania (na przykład zestaw znaków UCS-2), wywołanie nie powiedzie się z kodem przyczyny MQRC_SOURCE_INTEGER_ENC_ERROR lub MQRC_TARGET_INTEGER_ENC_ERROR w zależności od potrzeb.

Opcje kodowania są obsługiwane w następujących środowiskach: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows.

Opcja domyślna: Jeśli żadna z opcji opisanych wcześniej nie jest określona, można użyć następującej opcji:

MQDCC_BRAK

Nie określono żadnych opcji.

Wartość MQDCC_NONE jest zdefiniowana w dokumentacji programu pomocy. Nie jest zamierzone, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

SourceCCSID

Typ: MQLONG-wejście

Jest to identyfikator kodowanego zestawu znaków łańcucha wejściowego w produkcie *SourceBuffer*.

SourceLength

Typ: MQLONG-wejście

Jest to długość w bajtach łańcucha wejściowego w produkcie *SourceBuffer* . Wartość musi być równa zero lub większa.

SourceBuffer

Typ: MQCHARxSourceDługość-wejście

Jest to bufor zawierający łańcuch, który ma zostać przekształcony z jednego zestawu znaków na inny.

TargetCCSID

Typ: MQLONG-wejście

Jest to identyfikator kodowanego zestawu znaków zestawu znaków, do którego ma zostać przekształcona wartość *SourceBuffer* .

TargetLength

Typ: MQLONG-wejście

Jest to długość (w bajtach) buforu wyjściowego *TargetBuffer* ; Wartość musi być równa zero lub większa. Wartość ta może być mniejsza lub większa niż *SourceLength*.

TargetBuffer

Typ: MQCHARxTargetDługość-wyjście

Jest to łańcuch po przekształceniu go w zestaw znaków zdefiniowany przez produkt *TargetCCSID*. Przekształcony łańcuch może być krótszy lub dłuższy niż łańcuch bez konwersji. Parametr *DataLength* wskazuje liczbę zwróconych poprawnych bajtów.

DataLength

Typ: MQLONG-wyjście

Jest to długość łańcucha zwracanego w buforze wyjściowym *TargetBuffer*. Przekształcony łańcuch może być krótszy lub dłuższy niż łańcuch bez konwersji.

CompCode

Typ: MQLONG-wyjście

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

Błąd MQRC_DATA_LENGTH_ERROR

(2010, X'7DA') Parametr długości danych nie jest poprawny.

BŁĄD MQRC_DBCS_ERROR

(2150, X'866 ') Łańcuch DBCS nie jest poprawny.

BŁĄD MQRC_HCONN_ERROR

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

MQRC_SOURCE_BUFFER_ERROR

(2145, X'861 ') Parametr buforu źródłowego jest niepoprawny.

MQRC_SOURCE_CCSDID_ERROR, BŁĄD

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

MQRC_SOURCE_INTEGER_ENC_ERROR

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

MQRC_SOURCE_LENGTH_ERROR

(2143, X'85F') Parametr długości źródła nie jest poprawny.

MQRC_STORAGE_NOT_AVAILABLE

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

MQRC_TARGET_BUFFER_ERROR

(2146, X'862 ') Parametr buforu docelowego jest niepoprawny.

MQRC_TARGET_CCSDID_ERROR

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

MQRC_TARGET_INTEGER_ENC_ERROR

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

MQRC_TARGET_LENGTH_ERROR

(2144, X'860 ') Parametr długości docelowej nie jest poprawny.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQXCNCV (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
         TargetCCSID, TargetLength, TargetBuffer, &DataLength,
         &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;           /* Connection handle */
MQLONG   Options;         /* Options that control the action of
                           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 */
```

Deklaracja języka COBOL (tylko IBM i)

```
CALL 'MQXCNCV' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
```

```
SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** Connection handle
01 HCONN          PIC S9(9) BINARY.
** Options that control the action of 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.
```

Deklaracja asemblera S/390

```
CALL MQXCNCV, (HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,      X
               SOURCEBUFFER, TARGETCCSID, TARGETLENGTH, TARGETBUFFER, X
               DATALENGTH, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
OPTIONS	DS	F	Options that control the action of 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-wyjście konwersji danych

Wywołanie komendy MQ_DATA_CONV_EXIT opisuje parametry, które są przekazywane do wyjścia konwersji danych.

Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_DATA_CONV_EXIT (patrz uwaga o składni 11).

Ta definicja jest częścią interfejsu DCI (Data Conversion Interface) produktu IBM MQ, który jest jednym z interfejsów środowiska produktu IBM MQ.

Składnia

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parametry

DataConvExitParms

Typ: MQDXP-wejście/wyjście

Struktura ta zawiera informacje związane z wywoływaniem wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać wynik konwersji. Szczegółowe informacje na temat pól w tej strukturze można znaleźć w sekcji [“MQDXP-Dane-parametr wyjścia konwersji danych” na stronie 925](#).

MsgDesc

Typ: MQMD-input/output

Po wejściu do wyjścia jest to deskryptor komunikatu powiązany z danymi komunikatu przekazanego do wyjścia w parametrze *InBuffer*.

Uwaga: Parametr *MsgDesc* przekazany do wyjścia jest zawsze najnowszą wersją deskryptora MQMD obsługiwaną przez menedżer kolejek, który wywołuje wyjście. Jeśli wyjście ma być przenośne między różnymi środowiskami, wyjście sprawdzi pole *Version* w programie *MsgDesc*, aby sprawdzić, czy pola, do których ma dostęp wyjście, znajdują się w strukturze.

W następujących środowiskach wyjście jest przekazywane version-2 MQMD: AIX, HP-UX, IBM i, Solaris, Linux, Windows. We wszystkich innych środowiskach obsługujących wyjście konwersji danych wyjście jest przekazywane MQMD w wersji version-1.

Na wyjściu wyjście spowoduje zmianę wartości pól *Encoding* i *CodedCharSetId* na wartości żądane przez aplikację, jeśli konwersja się powiodła; zmiany te są odzwierciedlane z powrotem do aplikacji. Wszelkie inne zmiany wprowadzone przez wyjście do struktury są ignorowane; nie są one odzwierciedlane z powrotem do aplikacji.

Jeśli wyjście zwraca wartość MQXDR_OK w polu *ExitResponse* struktury MQDXP, ale nie powoduje zmiany pól *Encoding* lub *CodedCharSetId* w deskrypcorze komunikatu, menedżer kolejek zwraca dla tych pól wartości, których dane pola w strukturze MQDXP miały na wejściu do wyjścia.

InBufferDługość

Typ: MQLONG-wejście

Długość (w bajtach) *InBuffer*.

Jest to długość buforu wejściowego *InBufferi* określa liczbę bajtów, które mają być przetwarzane przez wyjście. *InBufferLength* jest mniejszą od długości danych komunikatu przed konwersją, a także długość buforu udostępnionego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

InBuffer

Wpisz: MQBYTEInBufferLength -wejście

Bufor zawierający nieprzekonwertowany komunikat.

Ten komunikat zawiera dane komunikatu przed konwersją. Jeśli wyjście nie jest w stanie przekształcić danych, menedżer kolejek zwraca zawartość tego buforu do aplikacji po zakończeniu wyjścia.

Uwaga: Wyjście nie powinno zmieniać *InBuffer*; Jeśli ten parametr zostanie zmieniony, wyniki nie zostaną zdefiniowane.

W języku programowania C ten parametr jest zdefiniowany jako wskaźnik-do-void.

OutBufferDługość

Typ: MQLONG-wejście

Długość (w bajtach) *OutBuffer*.

Jest to długość buforu wyjściowego *OutBuffer*, która jest taka sama, jak długość buforu udostępnianego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

OutBuffer

Typ: MQBYTEExOutBufferLength - dane wyjściowe

Bufor zawierający przekształcony komunikat.

W przypadku wyjścia z wyjścia, jeśli konwersja zakończyła się pomyślnie (zgodnie z wartością MQXDR_OK w polu *ExitResponse* parametru *DataConvExitParms*), program *OutBuffer* zawiera dane komunikatu, które mają zostać dostarczone do aplikacji, w żądanej reprezentacji. Jeśli konwersja nie powiodła się, wszystkie zmiany wprowadzone w tym buforze zostaną zignorowane.

W języku programowania C ten parametr jest zdefiniowany jako wskaźnik-do-void.

Użycie notatek

1. Wyjście konwersji danych jest to wyjście pisane przez użytkownika, które odbiera sterowanie podczas przetwarzania wywołania MQGET. Funkcja wykonywana przez wyjście konwersji danych jest zdefiniowana przez dostawcę wyjścia, jednak wyjście musi być zgodne z regułami opisanymi w tym miejscu oraz w powiązanej strukturze parametrów MQDXP.

Języki programowania, które mogą być używane na potrzeby wyjścia konwersji danych, są określane przez środowisko.

2. Wyjście jest wywoływane tylko wtedy, gdy *wszystkie* z następujących elementów są prawdziwe:

- Opcja MQGMO_CONVERT została określona w wywołaniu MQGET
- Pole *Format* w deskrytorze komunikatu nie ma wartości MQFMT_NONE.
- Komunikat nie znajduje się już w wymaganej reprezentacji, to znaczy jeden lub oba komunikaty *CodedCharSetId* i *Encoding* różnią się od wartości określonej przez aplikację w deskrytorze komunikatu dostarczonem w wywołaniu MQGET.
- Menedżer kolejek nie wykonał jeszcze pomyślnie konwersji
- Długość buforu aplikacji jest większa od zera
- Długość danych komunikatu jest większa od zera
- Do tej pory kod przyczyny w operacji MQGET to MQRC_NONE lub MQRC_TRUNCATED_MSG_ACCEPTED

3. Po zapisaniu wyjścia należy rozważyć kodowanie wyjścia w sposób, który umożliwia przekształcenie komunikatów, które zostały obcięte. Obcięte komunikaty mogą pojawić się w następujący sposób:

- Aplikacja odbierający udostępnia bufor, który jest mniejszy niż komunikat, ale określa opcję MQGMO_ACCEPT_TRUNCATED_MSG w wywołaniu MQGET.

W tym przypadku pole *Reason* w parametrze *DataConvExitParms* na wejściu do wyjścia ma wartość MQRC_TRUNCATED_MSG_ACCEPTED.

- Nadawca wiadomości obciął ją przed wysłaniem. Może się to zdarzyć w przypadku komunikatów raportu, na przykład (więcej informacji na ten temat zawiera sekcja ["Konwersja komunikatów raportu"](#) na stronie 925).

W tym przypadku pole *Reason* w parametrze *DataConvExitParms* na wejściu do wyjścia ma wartość MQRC_NONE (jeśli aplikacja odbierający udostępniła bufor, który był wystarczająco duży dla komunikatu).

Z tego powodu wartość pola *Reason* na wejściu do wyjścia nie może być zawsze używana do określenia, czy komunikat został obcięty.

Cechą wyróżniającą obciętą wiadomość jest to, że długość podana do wyjścia w parametrze *InBufferLength* jest *mniejsza niż* długość wynikająca z nazwy formatu zawartej w polu *Format* w deskrytorze komunikatu. Wyjście powinno więc sprawdzić wartość *InBufferLength* przed próbą przekształcenia danych; wyjście *nie powinno* zakładać, że pełna ilość danych implikowanych przez nazwę formatu została podana.

Jeśli wyjście *nie* zostało zapisane w celu przekształcenia obciętych komunikatów, a wartość *InBufferLength* jest mniejsza niż oczekiwana, wyjście zwróci wartość

MQXDR_CONVERSION_FAILED w polu *ExitResponse* parametru *DataConvExitParms*, a pola *CompCode* i *Reason* są ustawione na wartość MQCC_WARNING i MQRC_FORMAT_ERROR.

Jeśli wyjście *ma* zostało zapisane w celu przekształcenia obciętych komunikatów, wyjście spowoduje przekształcenie możliwie największej ilości danych (patrz uwaga w następnym użyciu), starając się nie próbować badać ani konwertować danych poza końcem *InBuffer*. Jeśli konwersja zakończy się pomyślnie, wyjście pozostawi pole *Reason* w parametrze *DataConvExitParms* bez zmian. Zwraca wartość MQRC_TRUNCATED_MSG_ACCEPTED, jeśli komunikat został obcięty przez menedżera kolejek odbiornika, i MQRC_NONE, jeśli komunikat został obcięty przez nadawcę komunikatu.

Możliwe jest również, że komunikat może rozwinąć *podczas* konwersji do punktu, w którym jest on większy niż *OutBuffer*. W takim przypadku wyjście musi zdecydować, czy komunikat ma zostać obcięty, a pole *AppOptions* w parametrze *DataConvExitParms* wskazuje, czy aplikacja odbierający określiła opcję MQGMO_ACCEPT_TRUNCATED_MSG.

4. Generalnie wszystkie dane w komunikacie dostarczonym do wyjścia w programie *InBuffer* są przekształcane, lub że żadne z nich nie jest. Wyjątkiem od tego jest jednak, jeśli komunikat jest obcinany przed konwersją lub podczas konwersji; w tym przypadku na końcu buforu może wystąpić niepełny element (na przykład: 1 bajt znaku dwubajtowego lub 3 bajty 4-bajtowej liczby całkowitej). W takiej sytuacji należy rozważyć pominięcie niekompletnego elementu i ustawienie nieużywanych bajtów w *OutBuffer* na wartości NULL. Jednak pełne elementy lub znaki w tablicy lub łańcuchu *powinny* być przekształcane.
5. Gdy wyjście jest wymagane po raz pierwszy, menedżer kolejek próbuje załadować obiekt o takiej samej nazwie, jak format (poza rozszerzeniami). Załadowany obiekt musi zawierać wyjście, które przetwarza komunikaty z tą nazwą formatu. Należy rozważyć wprowadzenie nazwy wyjścia i nazwy obiektu, który zawiera wyjście identyczne, chociaż nie wszystkie środowiska wymagają tego.
6. Nowa kopia wyjścia jest ładowana, gdy aplikacja próbuje pobrać pierwszy komunikat, który używa tego produktu *Format* od momentu połączenia aplikacji z menedżerem kolejek. W przypadku aplikacji CICS lub IMS oznacza to, kiedy podsystem CICS lub IMS jest połączony z menedżerem kolejek. Nowa kopia może być również załadowana w innym czasie, jeśli menedżer kolejek odrzucił wcześniej załadowaną kopię. Z tego powodu wyjście nie może próbować używać statycznej pamięci masowej do przekazywania informacji z jednego wywołania wyjścia do następnego-wyjście może być rozładowane między dwoma wywołaniami.
7. Jeśli istnieje wyjście podane przez użytkownika o tej samej nazwie co jeden z wbudowanych formatów obsługiwanych przez menedżer kolejek, wyjście podane przez użytkownika nie zastępuje wbudowanej procedury konwersji. Jedynymi okolicznościami, w których takie wyjście jest wywołane, są:
 - Jeśli wbudowana procedura konwersji nie może obsłużyć konwersji do lub z *CodedCharSetId* lub *Encoding* biorących udział, lub
 - Jeśli wbudowana procedura konwersji nie przekształci danych (na przykład, ponieważ istnieje pole lub znak, które nie mogą zostać przekształcone).
8. Zasięg wyjścia jest zależny od środowiska. Aby zminimalizować ryzyko wystąpienia starć z innymi formatami, należy wybrać nazwy produktu *Format*. Rozważ rozpoczęcie od znaków identyfikujących aplikację definiującą nazwę formatu.
9. Wyjście konwersji danych działa w środowisku takim jak program, który wywołał wywołanie MQGET; środowisko obejmuje przestrzeń adresową i profil użytkownika (jeśli ma to zastosowanie). Program może być agentem kanału komunikatów wysyłającym komunikaty do docelowego menedżera kolejek, który nie obsługuje konwersji komunikatów. Wyjście nie może naruszać integralności menedżera kolejek, ponieważ nie jest ono uruchamiane w środowisku menedżera kolejek.
10. Jedynym wywołaniem MQI, który może być używany przez wyjście, jest MQXCNCV; próba użycia innych wywołań MQI kończy się niepowodzeniem z kodem przyczyny MQRC_CALL_IN_PROGRESS lub innymi nieprzewidywalnymi błędami.
11. Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_DATA_CONV_EXIT. Jednak w języku programowania C podano typedef dla nazwy MQ_DATA_CONV_EXIT, a to może być używane do zadeklarowania wyjścia napisanego przez użytkownika, aby upewnić się, że parametry są poprawne.

Nazwa wyjścia musi być taka sama, jak nazwa formatu (nazwa zawarta w polu *Format* w strukturze MQMD), chociaż nie jest to wymagane we wszystkich środowiskach.

Poniższy przykład ilustruje sposób, w jaki wyjście, które przetwarza format MYFORMAT, może zostać zadeklarowane w języku programowania C:

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
    PMQDXP  pDataConvExitParms, /* Data-conversion exit parameter
                                block */
    PMQMD   pMsgDesc,           /* Message descriptor */
    MQLONG  InBufferLength,     /* Length in bytes of InBuffer */
    PMQVOID pInBuffer,          /* Buffer containing the unconverted
                                message */
    MQLONG  OutBufferLength,    /* Length in bytes of OutBuffer */
    PMQVOID pOutBuffer)         /* Buffer containing the converted
                                message */
{
    /* C language statements to convert message */
}
```

12. W systemie z/OS, jeśli wyjście funkcji API jest również wymuszone, jest wywoływane po wyjściu konwersji danych.

Wywołanie C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
          InBuffer, OutBufferLength, OutBuffer);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
MQDXP  DataConvExitParms; /* Data-conversion exit parameter block */
MQMD    MsgDesc;          /* Message descriptor */
MQLONG  InBufferLength;   /* Length in bytes of InBuffer */
MQBYTE  InBuffer[n];      /* Buffer containing the unconverted
                           message */
MQLONG  OutBufferLength;  /* Length in bytes of OutBuffer */
MQBYTE  OutBuffer[n];     /* Buffer containing the converted
                           message */
```

Deklaracja języka COBOL (tylko IBM i)

```
CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                      INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
** Data-conversion exit parameter block
01 DATACONVEXITPARMS.
   COPY CMQDXPV.
** Message descriptor
01 MSGDESC.
   COPY CMQMDV.
** Length in bytes of INBUFFER
01 INBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the unconverted message
01 INBUFFER PIC X(n).
** Length in bytes of OUTBUFFER
01 OUTBUFFERLENGTH PIC S9(9) BINARY.
** Buffer containing the converted message
01 OUTBUFFER PIC X(n).
```

Deklaracja asemblera System/390

```
CALL EXITNAME, (DATA CONVEXITPARMS, MSGDESC, INBUFFERLENGTH,      X
                INBUFFER, OUTBUFFERLENGTH, OUTBUFFER)
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

DATA CONVEXITPARMS	CMQDXPA	,	Data-conversion exit parameter block
MSGDESC	CMQMDA	,	Message descriptor
INBUFFERLENGTH	DS	F	Length in bytes of INBUFFER
INBUFFER	DS	CL(n)	Buffer containing the unconverted message
*			
OUTBUFFERLENGTH	DS	F	Length in bytes of OUTBUFFER
OUTBUFFER	DS	CL(n)	Buffer containing the converted message
*			

Właściwości określone jako elementy MQRFH2

Właściwości deskryptora innego niż message mogą być określone jako elementy w folderach nagłówka MQRFH2. Przegląd elementów MQRFH2, które są określone jako właściwości.

Zachowuje to kompatybilność z poprzednimi wersjami klientów IBM MQ JMS i XMS. W tej sekcji opisano sposób określania właściwości w nagłówkach MQRFH2.

Aby użyć elementów MQRFH2 jako właściwości, należy określić elementy zgodnie z opisem w sekcji [Korzystanie z produktu IBM MQ classes for Java](#). Te informacje uzupełniają informacje opisane w sekcji ["MQRFH2 -reguły i nagłówki formatowania 2"](#) na stronie 529.

Odwzorowywanie typów danych właściwości na typy danych MQRFH2

Ten temat zawiera informacje na temat typów właściwości komunikatu odwzorowanych na odpowiadające im typy danych MQRFH2.

Tabela 119. Obsługiwane typy danych MQRFH2	
Typ właściwości komunikatu	Typ danych MQRFH2
MQBYTE []	bin.hex
MQBOOL	boolean (boolowskie)
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	string (łańcuch)

Przyjmuje się, że dowolny element bez typu danych ma typ "string".

Typ danych MQRFH2 produktu int, oznaczający liczbę całkowitą nieokreśloną wielkość, jest traktowany tak, jakby był i8.

Wartość NULL jest wskazywana przez atrybut elementu `xsi:nil='true'`. Nie należy używać atrybutu `xsi:nil='false'` dla wartości innych niż NULL.

Na przykład następująca właściwość ma wartość NULL:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Właściwość typu byte lub łańcuch znaków może mieć pustą wartość. Jest to reprezentowane przez element MQRFH2 o wartości elementu o zerowej długości.

Na przykład następująca właściwość ma pustą wartość:

```
<EmptyProperty></EmptyProperty>
```

Obsługiwane foldery MQRFH2

Przegląd użycia pól deskryptora komunikatu jako właściwości.

Foldery <jms>, <mcd>, <mqext> i <usr> są opisane w sekcji Nagłówek MQRFH2 i JMS. Folder <usr> jest używany do transportów dowolnych właściwości zdefiniowanych przez aplikację produktu JMS, które są powiązane z komunikatem. Grupy nie są dozwolone w folderze <usr>.

Nagłówek MQRFH2 i JMS obsługuje następujące dodatkowe foldery:

- <mq>

Ten folder jest używany i zarezerwowany dla właściwości zdefiniowanych przez produkt MQ, które są używane przez produkt IBM MQ.

- <mq_usr>

Ten folder może być używany do transportów dowolnych właściwości zdefiniowanych przez aplikację, które nie są ujawnione jako właściwości zdefiniowane przez użytkownika produktu JMS, ponieważ właściwości te mogą nie spełniać wymagań właściwości JMS. Ten folder może zawierać grupy, których nie można użyć do folderu <usr>.

- Dowolny folder oznaczony atrybutem content= 'properties'.

Taki folder jest równoważny z folderem <mq_usr> w treści.

- <mmps>

Ten folder jest używany dla właściwości publikowania/subskrybowania produktu IBM MQ.

Produkt IBM MQ obsługuje również następujące foldery, które są już używane przez WAS/SIB:

- <sib>

Ten folder jest używany i zarezerwowany dla właściwości komunikatów systemowych WAS/SIB, które nie są ujawnione jako właściwości produktu JMS lub są odwzorowane na właściwości JMS_IBM_*, ale są ujawniane w aplikacjach WAS/SIB. Te właściwości zawierają właściwości ścieżek routingu zwrotnego i odwrotnego routingu.

Co najmniej niektóre nie mogą być prezentowane jako właściwości JMS, ponieważ są to tablice bajtów. Jeśli aplikacja doda właściwości do tego folderu, ta wartość zostanie zignorowana lub usunięta.

- <sib_usr>

Ten folder jest używany i zarezerwowany dla właściwości komunikatów użytkownika WAS/SIB, które nie mogą być prezentowane jako właściwości użytkownika produktu JMS, ponieważ nie są obsługiwane przez te właściwości. Są one ujawniane w aplikacjach WAS/SIB.

Są to właściwości użytkownika, które można uzyskać lub ustawić za pomocą interfejsu SIMessage, ale treść tablicy bajtów jest odwzorowana na wymaganą wartość właściwości.

Jeśli aplikacja IBM MQ zapisze dowolny element bin.hex do folderu, aplikacja prawdopodobnie otrzyma IOException, ponieważ nie jest to format oczekiwany do odtworzenia. W przypadku dodania elementu innego niż element bin.hex otrzymany jest ClassCastException.

Nie należy podejmować prób udostępniania właściwości WAS/SIB przy użyciu tego folderu. W tym celu należy zamiast tego użyć do tego celu folder <usr>.

- `<sib_context>`

Ten folder jest używany dla właściwości komunikatów systemu WAS/SIB, które nie są ujawnione dla aplikacji użytkownika WAS/SIB lub jako właściwości produktu JMS . Należą do nich właściwości zabezpieczeń i transakcyjne, które są używane dla usług Web Service i podobnych.

Aplikacja nie może dodawać właściwości do tego folderu.

- `<mqema>`

Ten folder został użyty przez WAS/SIB zamiast folderu `<mqext>` .

W nazwach folderów MQRFH2 rozróżniana jest wielkość liter.

Następujące foldery są zastrzeżone, w dowolnej mieszance małych lub wielkich liter:

- Dowolny folder poprzedzony przedrostkiem mq lub wmq ; zarezerwowane do użycia przez produkt IBM MQ.
- Każdy folder poprzedzony przedrostkiem sib ; zarezerwowane do użycia przez WAS/SIB.
- Foldery `<Root>` i `<Body>` ; zarezerwowane, ale nie używane.

Następujące foldery nie są rozpoznawane jako zawierające właściwości komunikatu:

- `<psc>`

Używany przez produkt IBM Integration Bus do przekazywania komunikatów komend publikowania/subskrypcji do brokera.

- `<pscr>`

Używany przez produkt IBM Integration Bus do przechowywania informacji z brokera w odpowiedzi na komunikaty komend publikowania/subskrypcji.

- Dowolny folder, który nie jest zdefiniowany przez produkt IBM, który nie jest oznaczony atrybutem `content='properties'` .

Nie należy określać `content='properties'` w folderach `<psc>` ani `<pscr>` . W takim przypadku foldery te są traktowane jako właściwości, a produkt IBM Integration Bus prawdopodobnie przestanie działać zgodnie z oczekiwaniami.

Jeśli aplikacja jest budowaniem komunikatów z właściwościami, w nagłówkach MQRFH2 , które mają być rozpoznawane jako nagłówek MQRFH2 zawierający właściwości, nagłówek musi znajdować się na liście nagłówków, które mogą być łańcuchowane w nagłówku komunikatu.

Wartość MQRFH2 może być poprzedzona dowolną liczbą nagłówków standardowych MQH lub MQCIH, MQDLH, MQIIH, MQTM, MQTMC2 lub MQXQH. Łańcuch lub tabela MQCFH kończy analizowanie, ponieważ nie mogą być połączone łańcuchami.

Istnieje możliwość, że komunikat będzie zawierał wiele nagłówków MQRFH2 wszystkich właściwości przesyłania komunikatów. Foldery o tej samej nazwie mogą współistnieć w różnych nagłówkach, o ile nie jest to inaczej ograniczone, na przykład przez WAS/SIB. Foldery są traktowane jako jeden folder logiczny, jeśli są one wszystkie w znaczących nagłówkach.

Podczas gdy foldery z istotnych nagłówków nie mogą być scalane z tymi folderami w nieistotnych nagłówkach, można scalić foldery o tej samej nazwie w znaczących nagłówkach, usuwając wszystkie właściwości powodujące konflikt. Aplikacje użytkownika nie mogą zależeć od układu właściwości w obrębie ich komunikatu.

Grupy MQRFH2 są analizowane pod kątem właściwości w folderach zdefiniowanych przez użytkownika, tj. nie w folderach `<wmq>`, `<jms>`, `<mcd>`, `<usr>`, `<mqext>`, `<sib>`, `<sib_usr>`, `<sib_context>` i `<mqema>` .

Grupy znajdujące się w folderach właściwości zdefiniowanych w produkcie IBM, z wyjątkiem folderów `<wmq>` i `<mq>` , są analizowane pod kątem właściwości.

Folder MQRFH2 nie może zawierać treści mieszanej. Folder lub grupa może zawierać albo grupy, albo właściwości, albo wartość, ale nie obie te wartości.

Segment komunikatu, który jest pierwszym lub kolejnym segmentem, nie może zawierać właściwości zdefiniowanych przez produkt IBM MQ innych niż te, które są zawarte w deskrytorze komunikatu. W związku z tym umieszczenie komunikatu zawierającego takie właściwości z zestawem MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED powoduje niepowodzenie operacji put z opcją MQRC_SEGMENTATION_NOT_ALLOWED.

Jednak grupy komunikatów mogą zawierać właściwości zdefiniowane w produkcie IBM MQ.

Generowanie nagłówków produktu MQRFH2

Jeśli program IBM MQ przekształca właściwości komunikatu w ich reprezentację MQRFH2, musi dodać MQRFH2 do komunikatu. Dodaje on MQRFH2 jako oddzielny nagłówek lub scala go z istniejącym nagłówkiem.

Generowanie nowych nagłówków MQRFH2 przez produkt IBM MQ może zakłócić istniejące nagłówki w komunikacie. Aplikacje, które analizują bufor komunikatów w nagłówkach, muszą mieć świadomość, że liczba i pozycja nagłówków w buforze mogą się zmieniać w pewnych okolicznościach. Program IBM MQ próbuje zminimalizować wpływ dodawania właściwości do komunikatu przez scalanie właściwości komunikatu do istniejącego nagłówka MQRFH2, w którym może on być wyświetlany. Podejmuje również próbę zminimalizowania wpływu poprzez wstawienie wygenerowanego MQRFH2 do stałej pozycji w stosunku do innych nagłówków w buforze komunikatów.

Wygenerowany nagłówek MQRFH2 jest umieszczany po MQMDi dowolnej liczbie nagłówków MQXQH, MQRFH, MQDLH, niezależnie od kolejności ich wprowadzenia. Wygenerowany nagłówek MQRFH2 jest umieszczany bezpośrednio przed pierwszym nagłówkiem, który nie jest nagłówkiem MQMD, MQXQH, MQDLH, lub MQRFH.

 W systemach z/OS wygenerowany nagłówek MQRFH2 jest tworzony w aplikacji CCSID aplikacji. Wartość ta jest zdefiniowana w następujący sposób:

- W przypadku zadań wsadowych LE za pomocą interfejsu DLL, CCSID jest CODESET powiązany z bieżącymi ustawieniami narodowymi w momencie wydawania **MQCONN** (wartość domyślna to 1047).
- W przypadku zadań wsadowych LE powiązanych z jednym z kodów pośredniczących MQ wsadowym CCSID jest CODESET powiązany z bieżącymi ustawieniami narodowymi w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku wsadowych aplikacji innych niż LE działających w wątku USS, CCSID jest wartością THLICCSID w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku innych aplikacji wsadowych CCSID jest CCSID menedżera kolejek.

W przypadku aplikacji LE ustawienia narodowe można zmienić za pomocą usługi wywoływalnej `setlocale()` / `CEESETL LE`. W przypadku aplikacji innych niż LE działających w wątkach USS, wartość THLICCSID może zostać zmieniona za pomocą makra odwzorowania USS **BPXYTHLI**.

Reguły scalania wygenerowanych MQRFH2

Poniższe reguły mają zastosowanie do scalania wygenerowanej partycji MQRFH2 z istniejącym MQRFH2. Wygenerowany nagłówek MQRFH2 jest scalany z istniejącym nagłówkiem MQRFH2, jeśli:

1. Istniejący produkt MQRFH2 znajduje się w tej samej pozycji IBM MQ, co powoduje umieszczenie wygenerowanego MQRFH2 lub wcześniejszego w łańcuchu nagłówka.
2. Identyfikator CCSID wygenerowanej właściwości jest taki sam, jak identyfikator NameValueCCSID istniejącej partycji MQRFH2.

W przeciwnym razie wygenerowany nagłówek jest umieszczany osobno w buforze, w położeniu opisanym wcześniej.

Reguły scalania folderów w istniejącym MQRFH2

Jeśli właściwości komunikatu zostaną scalone z istniejącym MQRFH2, wówczas istniejąca MQRFH2 jest skanowana w celu uzyskania folderów, które są zgodne z właściwościami komunikatu, i scala je. Jeśli

pasujący folder nie istnieje, do końca istniejących folderów zostanie dodany nowy folder. Jeśli zgodny folder istnieje, przeszukiwany jest folder. Wszystkie zgodne właściwości zostaną nadpisane. Wszystkie nowe elementy zostaną dodane na końcu folderu.

Ograniczenia folderu MQRFH2

Przegląd ograniczeń folderów w nagłówkach MQRFH2

Ograniczenia MQRFH2 mają zastosowanie do następujących folderów:

- Nazwy elementów w folderze <usr> nie mogą zaczynać się od przedrostka JMS ; takie nazwy właściwości są zarezerwowane do użycia przez produkt JMS i nie są poprawne dla właściwości zdefiniowanych przez użytkownika.

Taka nazwa elementu nie powoduje, że analizowanie obiektu MQRFH2 nie powiedzie się, ale nie jest dostępne dla interfejsów API właściwości komunikatu produktu IBM MQ .

- Nazwy elementów w folderze <usr> nie mogą być, w żadnej mieszance niższych lub wielkich liter, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS i ESCAPE. Nazwy te są zgodne ze słowami kluczowymi SQL i utrudniają analizowanie selektorów, ponieważ <usr> jest folderem domyślnym używanym, gdy dla konkretnej właściwości w selektorze nie jest określony żaden folder.

Taka nazwa elementu nie powoduje, że analizowanie obiektu MQRFH2 nie powiedzie się, ale nie jest dostępne dla interfejsów API właściwości komunikatu produktu IBM MQ .

- Model treści folderu <usr> jest następujący:
 - Jako nazwy elementu można użyć dowolnej poprawnej nazwy XML, pod warunkiem, że nie zawiera ona dwukropka.
 - Dozwolone są tylko elementy proste, a nie zagnieżdżone.
 - Wszystkie elementy przyjmują domyślny typ łańcucha, chyba że jest on modyfikowany przez atrybut `dt="xxx"` .
 - Wszystkie elementy są opcjonalne, ale powinny wystąpić nie więcej niż jeden raz w folderze.
- Nazwy elementów w dowolnym folderze uważanym za zawierające właściwości komunikatu nie mogą zawierać kropki (.) (znak Unicode U+002E), ponieważ jest on używany w nazwach właściwości w celu wskazania hierarchii.

Taka nazwa elementu nie powoduje, że analizowanie obiektu MQRFH2 nie powiedzie się, ale nie jest dostępne dla interfejsów API właściwości komunikatu produktu IBM MQ .

W ogólnym przypadku nagłówki MQRFH2 zawierające poprawne dane w stylu XML mogą być analizowane przez produkt IBM MQ bez niepowodzeń, chociaż niektóre elementy MQRFH2 nie są dostępne za pośrednictwem interfejsów API właściwości komunikatu produktu IBM MQ .

Konflikty nazw elementów MQRFH2

Przegląd konfliktów w nazwach elementów MQRFH2 .

Do właściwości komunikatu może zostać przyłączona tylko jedna wartość. Jeśli próba uzyskania dostępu do właściwości prowadzi do konfliktu wartości, to jedna z nich jest wybierana w preferowanej kolejności względem innej.

Składnia IBM MQ w celu uzyskania dostępu do elementów MQRFH2 pozwala na unikalność identyfikowania elementu, jeśli folder nie zawiera żadnych elementów o tej samej nazwie. Jeśli folder zawiera więcej niż jeden element o tej samej nazwie, wartość właściwości używanej przez tę właściwość jest najbardziej zbliżony do głowy komunikatu.

Ma to zastosowanie, jeśli dwa lub więcej folderów o tej samej nazwie znajduje się w różnych znaczących nagłówkach MQRFH2 w tym samym komunikacie.

Konflikt może spowodować, że wywołanie MQGET jest przetwarzane po dwukrotnym ustawieniu właściwości deskryptora innego niż deskryptor komunikatu: zarówno za pomocą wywołania MQSETMP, jak i bezpośrednio w nagłówku surowego nagłówka MQRFH2 .

W takim przypadku właściwość powiązana z komunikatem przy użyciu wywołania interfejsu API ma pierwszeństwo przed jednym w danych komunikatu, tj. tym, który znajduje się w surowej nagłówku MQRFH2 . Jeśli wystąpi konflikt, uznaje się, że jest on logicznie przed danymi komunikatu.

Odwzorowywanie nazw właściwości na folder MQRFH2 i nazwy elementów

Przegląd różnic między nazwami właściwości i nazwami elementów w nagłówku MQRFH2 .

W przypadku używania dowolnego z zdefiniowanych interfejsów API, które ostatecznie generują nagłówki MQRFH2 , w celu określenia właściwości komunikatu (na przykład MQ JMS) nazwa właściwości niekoniecznie musi być nazwą elementu w folderze MQRFH2 .

Dlatego odwzorowanie jest wykonywane z nazwy właściwości do elementu MQRFH2 , a w odwrotnym kierunku, biorąc pod uwagę zarówno nazwę folderu, który zawiera element, jak i nazwę elementu. Niektóre przykłady z produktu IBM MQ classes for JMS są już udokumentowane w produkcie [Korzystanie z Java](#).

Nazwa właściwości	Nazwa folderu MQRFH2	Nazwa elementu MQRFH2
JMSDestination	jms	Dst
JMSType	mcd	Type, Set, Fmt
xxx (zdefiniowany przez użytkownika, gdzie xxx nie rozpoczyna się od JMS)	usr	xxx

Dlatego też, gdy aplikacja JMS uzyskuje dostęp do właściwości JMSDestination , to jest ona odwzorowywać na element Dst w folderze <jms> .

Podczas określania właściwości jako elementów MQRFH2 produkt IBM MQ definiuje jego elementy w następujący sposób:

Nazwa właściwości	Nazwa folderu MQRFH2	Nazwa grupy MQRFH2	Nazwa elementu MQRFH2
<Property>	<usr>	nie dotyczy	<Property>
<folder>. <Property>	<folder>	nie dotyczy	<Property>
<folder>. <group>. <Property>	<folder>	<group>	<Property>

Na przykład, gdy aplikacja IBM MQ próbuje uzyskać dostęp do właściwości Property1 , to jest ona odwzorowywać na element Property1 w folderze <usr> . Właściwość wmq .Property2 jest odwzorowywać na właściwość Property2 w folderze <wmq> .

Jeśli nazwa właściwości zawiera więcej niż jeden element. Używana nazwa elementu MQRFH2 jest nazwą elementu po finale. Grupy znakowe i MQRFH2 są używane do tworzenia hierarchii. zagnieżdżone grupy MQRFH2 są dozwolone.

Nagłówek JMS i właściwości specyficzne dla dostawcy, które znajdują się w folderze MQRFH2 w folderach <mcd>, <jms> i <mext> , są dostępne w aplikacji IBM MQ przy użyciu nazw skróconej zdefiniowanych w sekcji [Korzystanie z produktu IBM MQ classes for Java](#) .

JMS uzyskuje dostęp do właściwości zdefiniowanych przez użytkownika z folderu <usr> . Aplikacja IBM MQ może użyć folderu <usr> dla właściwości aplikacji, jeśli jest ona akceptowalna dla właściwości, która ma być wyświetlana w aplikacjach produktu JMS jako jedna z jej właściwości zdefiniowanych przez użytkownika.

Jeśli nie jest to akceptowalne, należy wybrać inny folder. Folder <wmq_usr> jest udostępniany jako standardowe położenie dla takich właściwości innych niż JMS .

Aplikacje mogą określać i używać dowolnego folderu MQRFH2 z dobrze zdefiniowanym użyciem, a nie udokumentowany w programie [“Właściwości określone jako elementy MQRFH2”](#) na stronie 942 , jeśli użytkownik zauważy, że:

1. Folder może już być używany lub może być używany w przyszłości przez inną aplikację udostępniając niezdefiniowany dostęp do właściwości znajdujących się w nim. Patrz sekcja Nazwy właściwości dla sugerowanej konwencji nazewnictwa dla nazw właściwości.
2. Właściwości nie są dostępne dla poprzednich wersji klienta IBM MQ classes for JMS lub klienta XMS , które mogą uzyskiwać dostęp tylko do folderu <usr> w celu uzyskania właściwości zdefiniowanych przez użytkownika.
3. Folder musi być oznaczony atrybutem content o wartości ustawionej na properties (na przykład content= 'properties').

Produkt “MQSETMP-ustawienie właściwości komunikatu” na stronie 791 automatycznie dodaje ten atrybut zgodnie z wymaganiami. Ten atrybut nie może być dodawany do żadnego z folderów zdefiniowanych w produkcie IBM, na przykład <jms> i <usr>. Spowoduje to, że komunikat zostanie odrzucony przez klienta IBM MQ classes for JMS przed Wersja 7.0. z MessageFormatException.

Ponieważ folder <usr> jest domyślnym położeniem dla właściwości składni <Property> , aplikacji IBM MQ i aplikacji JMS w celu uzyskania dostępu do tej samej wartości właściwości zdefiniowanej przez użytkownika przy użyciu tej samej nazwy.

Nazwy zarezerwowanych folderów

Istnieje kilka zastrzeżonych nazw folderów. Nie można używać takich nazw, jak przedrostki folderów, na przykład Root . Property1 nie ma dostępu do poprawnej właściwości, ponieważ Root jest zastrzeżony. Poniższa lista zawiera zastrzeżone nazwy folderów:

- Główny element
- Treść
- Właściwości
- Środowisko
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- Środowisko InputLocal
- Lista InputDestination
- Lista InputException
- OutputRoot
- Środowisko OutputLocal
- Lista OutputDestination
- Lista OutputException

Odzworowywanie pól deskryptora właściwości na nagłówki MQRFH2

Gdy właściwość jest przekształcana w element MQRFH2 , do określenia znaczących pól deskryptora właściwości są używane następujące atrybuty elementu: W ten sposób opisano, w jaki sposób pola MQPD są przekształcane w atrybuty elementu MQRFH2 .

Obsługa

Pole deskryptora właściwości obsługi jest podzielone na trzy atrybuty elementu.

- Atrybut elementu **sr** określa wartości w masce bitowej MQPD_REJECT_UNSUP_MASK.
- Atrybut elementu **sa** określa wartości w masce bitowej MQPD_ACCEPT_UNSUP_MASK.

- Atrybut elementu **sx** określa wartości w masce bitowej MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Te atrybuty elementów są poprawne tylko w folderze < mq> i są ignorowane, jeśli są ustawione na elementach w innych folderach zawierających właściwości.

Wartość wsparcia	Atrybut elementu MQRFH2	Wartość atrybutu MQRFH2
MQPD_SUPPORT_OPTIONAL	sa	opcjonalne Jest to wartość domyślna.
MQPD_SUPPORT_REQUIRED	SR	wymagane
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	lokalne

Kontekst

Atrybut elementu **context** służy do wskazywania kontekstu komunikatu, do którego należy właściwość. Należy użyć tylko jednej wartości. Ten atrybut elementu jest poprawny dla właściwości w dowolnym folderze zawierającym właściwości.

Wartość kontekstu	Wartość atrybutu MQRFH2
MQPD_NO_CONTEXT	brak Jest to wartość domyślna.
MQPD_USER_CONTEXT	użytkownik

CopyOptions

Atrybut elementu **copy** służy do wskazywania komunikatów, do których ma zostać skopiowana właściwość. Dopuszczalna jest więcej niż jedna wartość; oddziel wiele wartości przecinkiem. Na przykład: **copy='reply'** i **copy='publish,report'** są poprawne. Ten atrybut elementu jest poprawny dla właściwości w dowolnym folderze zawierającym właściwości.

Uwaga: W definicji atrybutu używane są pojedyncze znaki cudzysłowu lub podwójne cudzysłowy, na przykład **copy='reply'** lub **copy="report"**.

Wartość CopyOption	Wartość atrybutu MQRFH2
MQPD_COPY_FORWARD	postępująca
MQPD_COPY_REPLY (ODPOWIEDŹ)	reply
RAPORT MQPD_COPY_REPORT	raport
MQPD_COPY_PUBLISH	publikować
MQPD_COPY_ALL	Wszystkie Nie należy określać tej wartości przy użyciu żadnej innej wartości. W przypadku użycia z inną wartością ma ona pierwszeństwo przed dowolną wartością z wyjątkiem none .
MQPD_COPY_DEFAULT	default Jest to wartość domyślna. Jest on równoważny z podaniem trzech wartości: MQCOPY_FORWARD, MQCOPY_REPORT i MQCOPY_PUBLISH. Nie należy określać tej wartości przy użyciu żadnej innej wartości.

Wartość CopyOption	Wartość atrybutu MQRFH2
MQPD_COPY_BRAK	brak Nie należy określać tej wartości przy użyciu żadnej innej wartości. W przypadku użycia z inną wartością ma to pierwszeństwo.

Ograniczenia dotyczące folderu < mq> MQRFH2

Gdy komunikat jest umieszczany w kolejce, jest on przeszukiwany pod kątem folderu < mq>, dzięki czemu komunikat może być przetwarzany zgodnie z jego właściwościami zdefiniowanymi przez produkt MQ. Aby umożliwić sprawne analizowanie właściwości zdefiniowanych przez produkt MQ, do folderu mają zastosowanie następujące ograniczenia:

- Tylko właściwości w pierwszym znaczącym folderze < mq> w komunikacie są zachowane przez produkt MQ; właściwości w dowolnym innym folderze < mq> w komunikacie są ignorowane.
- Jeśli folder znajduje się w UTF-8, w folderze dozwolone są tylko znaki jednobajtowe UTF-8 . Wielobajtowy znak w folderze, może spowodować niepowodzenie analizowania, a komunikat zostanie odrzucony.
- Grupy MQRFH2 nie należy uwzględniać w folderze < mq>. Obecność znaku Unicode U+003C w wartości właściwości spowoduje, że komunikat zostanie odrzucony.
- W folderze nie należy używać łańcuchów zmiany znaczenia. Łańcuch zmiany znaczenia jest traktowany jako rzeczywista wartość elementu.
- Tylko znak Unicode U+0020 jest traktowany jako biały znak w folderze. Wszystkie inne znaki są traktowane jako znaczące i mogą spowodować niepowodzenie analizowania folderu, a komunikat do odrzucenia.

Jeśli analizowanie folderu < mq> nie powiedzie się lub jeśli folder nie będzie obserwował tych ograniczeń, komunikat zostanie odrzucony z opcją CompCode **MQCC_FAILED** i przyczyną **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Nagłówki MQRFH2 nie są poprawne.

W czasie przetwarzania wywołania MQPUT, MQPUT1 lub MQGET, może wystąpić częściowe analizowanie wszystkich nagłówków MQRFH2 w komunikacie w celu sprawdzenia, które foldery są dołączone, a także określenie, czy foldery zawierają właściwości. Przegląd nagłówków MQRFH2 , które nie są poprawne.

Jeśli częściowa analiza komunikatu nie może zostać zakończona pomyślnie, ponieważ struktura nie jest poprawna, na przykład pole StructLength jest zbyt małe, a następnie:

- Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_RFH_ERROR, jeśli można określić, że aplikacja zawiera opcję IBM MQ w wersji 7, dzięki czemu istniejące aplikacje nie będą kończyć się niepowodzeniem.
- Wywołanie MQGET zostało pomyślnie zwrócone, a komunikat MQRFH2 zawierający błąd jest zwracany w udostępnionym buforze.

Jeśli częściowe analizowanie nie powiedzie się, ponieważ nie można wykryć, czy dany folder zawiera właściwości, czy nie, na przykład folder zaczyna się od <<jms, dlatego analizowanie nie powiedzie się, zanim zostanie określona nazwa folderu, a następnie:

- Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_RFH_FORMAT_ERROR, jeśli można określić, że aplikacja zawiera opcję IBM MQ w wersji 7, dzięki czemu istniejące aplikacje nie będą kończyć się niepowodzeniem.
- Wywołanie MQGET zostało pomyślnie zwrócone, a komunikat MQRFH2 zawierający błąd jest zwracany w udostępnionym buforze.
- Podczas pracy wewnętrznie w menedżerze kolejek komunikat nie jest odrzucany ze względu na źle sformatowany folder, ale folder jest zawsze traktowany tak, jakby żadne właściwości nie znajdowały się w nim w nim zawarte.

Komunikat może przepływać przez sieć menedżera kolejek z folderem zawierającym taki błąd składniowy, ale nigdy nie jest analizowany ani wykrywany, podczas gdy co najmniej jeden folder w komunikacie jest następujący:

- Ważne
- Pomyślnie przeanalizowano
- Używane podczas przetwarzania komunikatu

W związku z tym wykrywanie nie jest gwarantowane.

Jeśli jedna z aplikacji korzysta z produktu “MQSETMP-ustawienie właściwości komunikatu” na stronie 791 lub MQINQMP w celu uzyskania dostępu do właściwości, a to powoduje, że folder MQRFH2 zostanie w pełni przeanalizowany, wykrycie błędu, który nie może zostać zakończony, jest wskazywane przez odpowiedni kod powrotu do wywołania API. Żadne właściwości w folderze nie są dostępne dla aplikacji.

Jeśli podejmowana jest próba pełnego przeanalizowania folderu MQRFH2, a analizator składni znajdzie nierozpoznane atrybuty elementu lub nierozpoznany typ danych, analizowanie jest kontynuowane i pomyślnie zakończone bez żadnych ostrzeżeń; nie stanowi to błędu analizowania.

konwersja stron kodowych

W tej sekcji opisano nazwy zestawów kodowych i identyfikatory CCSID, język narodowy, konwersję produktu z/OS, konwersję produktu IBM i oraz obsługę konwersji Unicode.

Każda sekcja w języku narodowym zawiera następujące informacje:

- Obsługiwane są rodzime identyfikatory CCSID
- Konwersje stron kodowych, które **nie** są obsługiwane

W informacjach używane są następujące terminy:

-8

Wskazuje HP-UX, że identyfikator CCSID jest przeznaczony dla zdefiniowanego zestawu kodowego HP-UX *roman8*

AIX

Oznacza IBM MQ for AIX

HP-UX

Oznacza IBM MQ for HP-UX

Linux

Indicates IBM MQ for Linux for Intel and IBM MQ for Linux for zSeries

HP Integrity NonStop Server

Oznacza IBM WebSphere MQ for HP Integrity NonStop Server

OS/400

Oznacza IBM MQ for IBM i

Solaris

Oznacza IBM MQ for Solaris

Windows

Oznacza IBM MQ for Windows

z/OS

Oznacza IBM MQ for z/OS

Wartość domyślna konwersji danych służy do konwersji, która ma być wykonywana w systemie docelowym (odbierającym).

Jeśli produkt źródłowy obsługuje konwersję, można skonfigurować kanał i wymieniać dane, ustawiając atrybut kanału CONVERT na wartość YES w źródle.

Uwaga:

1. Konwersja informacji o IBM MQ MQI client odbywa się na serwerze, więc serwer musi obsługiwać konwersję z identyfikatora CCSID klienta na identyfikator CCSID serwera.
2. Konwersja może obejmować obsługę dodaną przez CSD/PTF do najnowszej wersji produktu IBM MQ. Sprawdź zawartość najnowszego poziomu serwisowego, aby sprawdzić, czy konieczne jest zainstalowanie poprawki CSD/PTF w celu włączenia tej konwersji.
3. Identyfikator CCSID menedżera kolejek produktu IBM MQ musi być mieszany lub SBSCS.
4. Niektóre identyfikatory CCSID, na przykład 850 w systemie AIX, które nie są obsługiwane przez system operacyjny, nadal mogą być używane przez aplikację, a także mogą być ustawione jako identyfikator CCSID menedżera kolejek produktu IBM MQ . Jest to dozwolone tylko w celu zapewnienia kompatybilności wstecznej, a konwersja nie powiedzie się, jeśli odpowiednie tabele konwersji nie zostaną zainstalowane.

Patrz Tabela 120 na stronie 952 , aby uzyskać wzajemne odniesienie między niektórymi numerami CCSID i niektórymi branżami nazw zestawów kodowych.

Nazwy zestawów kodowych i identyfikatory CCSID

Produkt IBM MQ for z/OS zapewnia większą konwersję niż jest to wymienione w tabelach specyficznych dla języka.

Tabela 120. Nazwy zestawów kodowych i identyfikatory CCSID	
Nazwy zestawów kodowych	Identyfikatory CCSID
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (euro)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
PCK	943
GBk	1386
koi8-r	878

 Pełna lista dostępnych konwersji jest wyświetlana w produkcie [Tabela 121 na stronie 978](#).

Języki narodowe

Te informacje zawierają języki obsługiwane przez produkt IBM MQ.

Języki obsługiwane przez produkt IBM MQ są następujące:

- Angielski (Stany Zjednoczone)-patrz temat [“angielski \(USA\)” na stronie 953](#)
- Niemiecki-patrz temat [“niemiecki” na stronie 954](#)
- Duński i norweski-patrz temat [“Duński i norweski” na stronie 954](#)
- Fiński i szwedzki-patrz temat [“fiński i szwedzki” na stronie 955](#)
- Włoski-patrz temat [“włoski” na stronie 956](#)
- Hiszpański-patrz temat [“hiszpański” na stronie 957](#)
- Angielski/Gaelic w Wielkiej Brytanii-patrz temat [“angielski \(Wielka Brytania\) /Gaelic” na stronie 957](#)
- Francuski-patrz temat [“francuski” na stronie 958](#)
- Wielojęzyczny-patrz temat [“Wielojęzyczne” na stronie 958](#)
- Portugalski-patrz temat [“portugalski” na stronie 959](#)
- Islandzki-patrz temat [“islandzki” na stronie 960](#)
- Języki wschodnioeuropejskie-patrz temat [“Języki wschodnioeuropejskie” na stronie 961](#)
- Cyrylica-patrz temat [“cyrylica” na stronie 962](#)
- Estoński-patrz temat [“estoński” na stronie 962](#)
- Łotewski i litewski-patrz temat [“łotewski i litewski” na stronie 963](#)
- Ukraiński-patrz temat [“ukraiński” na stronie 965](#)
- Grecki-patrz temat [“grecki” na stronie 965](#)
- Turecki-patrz temat [“turecki” na stronie 966](#)
- Hebrajski-patrz temat [“hebrajski” na stronie 966](#)
- Farsi-patrz temat [“Farsi” na stronie 968](#)
- Urdu-patrz temat [“urdu” na stronie 968](#)
- Tajski-patrz temat [“tajski” na stronie 969](#)
- Lao-patrz temat [“laotański” na stronie 969](#)
- Wietnamski-patrz temat [“wietnamski” na stronie 969](#)
- Japoński Latin SBCS-patrz temat [“Japońskie łacińskie SBCS” na stronie 970](#)
- Japońska Katakana SBCS-patrz temat [“Japońska Katakana SBCS” na stronie 971](#)
- Japoński Kanji/Latin Mieszane-patrz temat [“Japoński Kanji/Latin Mieszane” na stronie 973](#)
- Japoński Kanji/Katakana Mieszane-patrz temat [“Japoński Kanji/Katakana Mieszane” na stronie 974](#)
- Koreański-patrz temat [“koreański” na stronie 975](#)
- Chiński uproszczony-patrz temat [“chiński uproszczony” na stronie 976](#)
- Chiński tradycyjny-patrz temat [“chiński tradycyjny” na stronie 977](#)

angielski (USA)

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka angielskiego (Stany Zjednoczone).

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka angielskiego w Stanach Zjednoczonych na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	37, 924, 1140
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 1252, 5348, 858

Platforma	Rodzime identyfikatory CCSID
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

37

Nie konwertuje na strony kodowe 923, 858

924

Nie konwertuje na strony kodowe 437, 858, 1051, 1140, 1252, 1275, 5348

1140

Nie konwertuje na strony kodowe 924, 1051, 1275

niemiecki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka niemieckiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka niemieckiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	273, 924, 1141
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

273

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje na strony kodowe 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141

Nie konwertuje na strony kodowe 924, 1051, 1275

Duński i norweski

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka duńskiego i norweskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka duńskiego i norweskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	277, 924, 1142
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 865, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

277

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje na strony kodowe 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142

Nie konwertuje na strony kodowe 924, 865, 1051, 1275

AIX

Strona kodowa:

819

Nie konwertuje na stronę kodową 865

HP-UX

Strona kodowa:

1051

Nie konwertuje na stronę kodową 865

Windows

Strona kodowa:

865

Nie konwertuje na strony kodowe 1051, 1275

fiński i szwedzki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla fińskiego i szwedzkiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla fińskiego i szwedzkiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	278, 924, 1143
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 865, 1252, 5348

Platforma	Rodzime identyfikatory CCSID
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

278

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje do stron kodowych 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143

Nie konwertuje na strony kodowe 865, 924, 1051, 1275

AIX

Strona kodowa:

819

Nie konwertuje na stronę kodową 865

850

Nie konwertuje na stronę kodową 865

HP-UX

Strona kodowa:

1051

Nie konwertuje na stronę kodową 865

Windows

Strona kodowa:

865

Nie konwertuje na strony kodowe 1051, 1275

włoski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka włoskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka włoskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

280

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje na strony kodowe 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144

Nie konwertuje na strony kodowe 924, 1051, 1275

hiszpański

Szczegółowe informacje o identyfikatorach CCSID i CCSID dla języka hiszpańskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka hiszpańskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

284

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje na strony kodowe 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145

Nie konwertuje na strony kodowe 924, 1051, 1275

angielski (Wielka Brytania) /Gaelic

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla brytyjskiego angielskiego/gaelickiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla brytyjskiego angielskiego/gaelickiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	285, 924, 1146
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348

Platforma	Rodzime identyfikatory CCSID
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

285

Nie konwertuje na strony kodowe 858, 923, 924, 1275

924

Nie konwertuje na strony kodowe 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146

Nie konwertuje na strony kodowe 924, 1051, 1275

francuski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka francuskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka francuskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

297

Nie konwertuje na strony kodowe 858, 923, 924, 1275, 5348

924

Nie konwertuje na strony kodowe 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147

Nie konwertuje na strony kodowe 924, 1051, 1275

Wielojęzyczne

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla systemu wielojęzycznego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla konwersji wielojęzycznej na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

500

Nie konwertuje na strony kodowe 858, 923

924

Nie konwertuje na strony kodowe 437, 858, 1051, 1148, 1252, 1275, 5348

1148

Nie konwertuje na strony kodowe 924, 1051, 1275

portugalski

Szczegółowe informacje o identyfikatorach CCSID i CCSID dla języka portugalskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka portugalskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i	37, 500, 924, 1140
z/OS	500, 924, 1140
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 860, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

37

Nie konwertuje na strony kodowe 858, 923, 1275

500

Nie konwertuje na strony kodowe 858, 923, 1275

924

Nie konwertuje na strony kodowe 858, 860, 1051, 1140, 1252, 1275, 5348

1140

Nie konwertuje na strony kodowe 860, 924, 1051, 1275

HP-UX

Strona kodowa:

1051

Nie konwertuje na stronę kodową 860

Windows

Strona kodowa:

860

Nie konwertuje na strony kodowe 1051, 1275

islandzki

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka islandzkiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka islandzkiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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, Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

871

Nie konwertuje na strony kodowe 858, 923, 924, 1275, 5348

924

Nie konwertuje na strony kodowe 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149

Nie konwertuje na strony kodowe 924, 1051, 1275

HP-UX

Strona kodowa:

1051

Nie konwertuje na stronę kodową 861

Windows

Strona kodowa:

861

Nie konwertuje na strony kodowe 1051, 1275

Języki wschodnioeuropejskie

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języków wschodnioeuropejskich. Do typowych języków używających tych identyfikatorów CCSID należą: albański, chorwacki, czeski, węgierski, polski, rumuński, serbski, słowacki, słoweński.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języków wschodnioeuropejskich na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	870, 1153
Windows	852, 1250, 5346, 9044
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	912
Wschodnioeuropejski klient Apple	1282
Rumuński klient Apple	1285
Chorwacki klient Apple	1284

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

870

Nie konwertuje na strony kodowe 1284, 1285

1153

Nie konwertuje na strony kodowe 1250, 1284, 1285

IBM i

Strona kodowa:

870

Nie konwertuje na strony kodowe 1284, 1285, 5346, 9044

1153

Nie konwertuje na strony kodowe 1282, 1284, 1285, 5346, 9044

HP-UX, Solaris, Linux

Strona kodowa:

912

Nie konwertuje na strony kodowe 1284, 1285

HP Integrity NonStop Server

Strona kodowa:

912

Nie konwertuje na strony kodowe 1153, 1284, 1285, 9044

Windows

Strona kodowa:

852

Nie konwertuje na strony kodowe 1284, 1285

1250

Nie konwertuje na strony kodowe 1284, 1285

9044

Nie konwertuje na strony kodowe 912, 1282, 1284, 1285

cyrilica

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla cyrylicy. Typowe języki używające tych CCSID to: Belarussian, bułgarski, macedoński, rosyjski i serbski.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla platformy Cyrillic na obsługiwanych platformach:

Platforma	Rodzime identyfikatory 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
Klient Apple	1283

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

880

Nie konwertuje na strony kodowe 855, 866, 878, 1131, 5347

1025

Nie konwertuje na strony kodowe 878, 5347

Windows

Strona kodowa:

855

Nie konwertuje na stronę kodową 1131

866

Nie konwertuje na stronę kodową 1131

1131

Nie konwertuje na strony kodowe 855, 866, 880, 1283

estoński

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka estońskiego.

Poniższa tabela przedstawia rodzime identyfikatory CCSID dla języka estońskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1122, 1157
Windows	902, 922, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	902, 922
HP Integrity NonStop Server	922

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

1122

Nie konwertuje na strony kodowe 902, 1157, 9449

1157

Nie konwertuje na strony kodowe 922, 1122, 1257, 9449

IBM i

Strona kodowa:

1122

Nie konwertuje na strony kodowe 902, 5353, 9449

1157

Nie konwertuje na strony kodowe 922, 5353, 9449

HP-UX, Solaris, Linux

Strona kodowa:

902

Nie konwertuje na strony kodowe 922, 1122, 9449

922

Nie konwertuje na strony kodowe 902, 1157, 9449

Windows

Strona kodowa:

5353

Nie konwertuje na stronę kodową 9449

9449

Nie konwertuje na strony kodowe 902, 922, 1122, 1157, 1257, 5353

902

Nie konwertuje na strony kodowe 922, 1122, 9449

HP Integrity NonStop Server

Strona kodowa:

922

Nie konwertuje na strony kodowe 902, 1157, 9449

łotewski i litewski

Szczegóły dotyczące CCSID i konwersji CCSID dla łotewskiego i litewskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID języka totewskiego i litewskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1112, 1156
Windows	901, 921, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	901, 921
HP Integrity NonStop Server,	921

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

1112

Nie konwertuje na strony kodowe 901, 1156, 9449

1156

Nie konwertuje na strony kodowe 901, 1156, 9449

IBM i

Strona kodowa:

1112

Nie konwertuje na stronę kodową 5353

1153

Nie konwertuje na strony kodowe 921, 5353, 9449

HP-UX, Solaris, Linux

Strona kodowa:

902

Nie konwertuje na strony kodowe 921, 1112, 1257, 9449

921

Nie konwertuje na strony kodowe 901, 1156, 9449

Windows

Strona kodowa:

901

Nie konwertuje na strony kodowe 921, 1112, 1257, 9449

5355

Nie konwertuje na stronę kodową 9449

9449

Nie konwertuje na strony kodowe 901, 921, 1112, 1156, 1257

HP Integrity NonStop Server

Strona kodowa:

921

Nie konwertuje na strony kodowe 901, 1156, 9449

ukraiński

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka ukraińskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla Ukraińców na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1123
Windows	1124, 1125, 1251, 5347
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1124

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

1123

Nie konwertuje na stronę kodową 5347

HP-UX

Strona kodowa:

1124

Nie konwertuje na stronę kodową 5347

Windows

Strona kodowa:

1125

Nie konwertuje na stronę kodową 1123

grecki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka greckiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka greckiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	875
HP-UX	813 (patrz uwaga)
Windows	869, 1253, 5349
AIX, NCR, HP Integrity NonStop Server, Solaris, Linux	813
Klient Apple	1280
Klient DOS	737

Uwaga: W systemie HP-UX obsługiwany jest tylko zestaw kodowy ISO. Własny zestaw kodowy greek8 produktu HP-UX nie ma zarejestrowanego identyfikatora CCSID i nie jest obsługiwany.

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID, własnymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

875

Nie konwertuje na stronę kodową 5349

Windows

Strona kodowa:

1253

Nie konwertuje na stronę kodową 737

5349

Nie konwertuje na stronę kodową 737

turecki

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka tureckiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka tureckiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1026
HP-UX	920 (patrz uwaga)
Windows	857, 1254, 5350
AIX, HP Integrity NonStop Server, Solaris, Linux	920
Klient Apple	1281

Uwaga: W systemie HP-UX obsługiwany jest tylko zestaw kodowy ISO. Własny zestaw kodowy turkish8 produktu HP-UX nie ma zarejestrowanego identyfikatora CCSID i nie jest obsługiwany.

Wszystkie platformy inne niż klienckie obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

1026

Nie konwertuje na stronę kodową 5350

hebrajski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka hebrajskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka hebrajskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
z/OS	424, 803, 4899, 12712
IBM i	424
AIX	916, 9048
HP-UX	916 (patrz uwaga)
Windows	1255, 5351
HP Integrity NonStop Server, Solaris, Linux	916

Uwaga: W systemie HP-UX obsługiwany jest tylko zestaw kodowy ISO. Własny zestaw kodowy greek8 produktu HP-UX nie ma zarejestrowanego identyfikatora CCSID i nie jest obsługiwany.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

424

Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

803

Nie konwertuje na strony kodowe 867, 4899, 5351, 9048, 12712

4899

Nie konwertuje na strony kodowe 424, 803, 856, 862, 916, 1255

12712

Nie konwertuje na strony kodowe 424, 803, 856, 916, 1255

IBM i

Strona kodowa:

424

Nie konwertuje na strony kodowe 803, 867, 4899, 5351, 9048, 12712

Strona kodowa 424 również konwertuje do i z CCSID 4952, który jest wariantem 856.

AIX

Strona kodowa:

916

Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

9048

Nie konwertuje na strony kodowe 424, 803, 856, 862, 916, 1255

Windows

Strona kodowa:

1255

Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

5351

Nie konwertuje na stronę kodową 803

arabski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka arabskiego

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka arabskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	420
AIX	1046, 1089
HP-UX	1089 (patrz uwaga)
Windows	720, 864, 1256, 5352

Platforma	Rodzime identyfikatory CCSID
HP Integrity NonStop Server, Solaris, Linux	1089

Uwaga: W systemie HP-UX obsługiwany jest tylko zestaw kodowy ISO. Własny zestaw kodowy arabic8 produktu HP-UX nie ma zarejestrowanego identyfikatora CCSID i nie jest obsługiwany.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

420

Nie konwertuje na stronę kodową 5352

HP-UX, Solaris, Linux, HP Integrity NonStop Server, Tru64

Strona kodowa:

1089

Nie konwertuje na stronę kodową 720

Windows

Strona kodowa:

720

Nie konwertuje na strony kodowe 1089, 5352

5352

Nie konwertuje na stronę kodową 720

Farsi

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla Farsi.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla produktu Farsi na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1097
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1098 (patrz uwaga)

Uwaga: Rodzimy identyfikator CCSID dla tych platform nie został zestandaryzowany i może ulec zmianie.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

urdu

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla urdu.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla serwera Urdu na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	918
Windows	868

Platforma	Rodzime identyfikatory CCSID
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1006

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

918

Nie konwertuje na stronę kodową 1006

tajski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka tajskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka tajskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	838
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	874 (patrz uwaga)

Uwaga: Rodzimy identyfikator CCSID dla tych platform nie został zestandaryzowany i może ulec zmianie.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

laotański

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla Lao.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla Lao na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1132
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1133

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

wietnamski

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka wietnamskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka wietnamskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1130
Windows	1258, 5354
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1129

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

1130

Nie konwertuje na strony kodowe 1129, 5354

Japońskie łacińskie SBCS

Szczegóły dotyczące CCSID i konwersji CCSID dla japońskiego łacińskiego SBCS.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla japońskiego łacińskiego SBCS na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1027
AIX	932, 5050, 33722 (patrz uwaga 1)
Windows	932, 943 (patrz uwagi 2 i 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050
HP-UX	Nie znana

Uwaga:

1. Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
2. Produkt Windows NT używa strony kodowej 932, ale najlepiej jest to przedstawić przy użyciu identyfikatora CCSID 943. Jednak nie wszystkie platformy produktu IBM MQ obsługują ten identyfikator CCSID.

W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można dokonać zmiany w pliku `../conv/table/ccsid.tbl`, która zmienia identyfikator CCSID używany na 943.

3. Produkt IBM MQ nie obsługuje stron kodowych w oparciu o standard JIS X 0213 (JIS2004).

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

1027

Nie konwertuje na strony kodowe 932, 942, 943, 954, 5050, 33722

IBM i

Strona kodowa:

1027

Nie konwertuje na stronę kodową 932

AIX

Strona kodowa:

932

Nie konwertuje na stronę kodową 1027

5050

Nie konwertuje na stronę kodową 1027

33722

Nie konwertuje na stronę kodową 1027

Linux

Strona kodowa:

943

Nie konwertuje na stronę kodową 1027

5050

Nie konwertuje na stronę kodową 1027

Solaris

Strona kodowa:

943

Nie konwertuje na stronę kodową 1027

5050

Nie konwertuje na stronę kodową 1027

HP Integrity NonStop Server

Strona kodowa:

943

Nie konwertuje na stronę kodową 1027

5050

Nie konwertuje na stronę kodową 1027

Japońska Katakana SBCS

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla japońskiej firmy Katakana SBCS.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla japońskiego systemu Katakana SBCS na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	290
HP-UX	897
AIX	932, 5050, 33722 (patrz uwaga 1)
Windows	932, 943 (patrz uwagi 2 i 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Uwaga:

1. Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
2. Produkt Windows NT używa strony kodowej 932, ale najlepiej jest to przedstawić przy użyciu identyfikatora CCSID 943. Jednak nie wszystkie platformy produktu IBM MQ obsługują ten identyfikator CCSID.

W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można dokonać zmiany w pliku `../conv/table/ccsid.tbl`, która zmienia identyfikator CCSID używany na 943.

3. Produkt IBM MQ nie obsługuje stron kodowych w oparciu o standard JIS X 0213 (JIS2004).

4. Oprócz wcześniejszych konwersji produkty IBM MQ w systemach AIX, HP-UX, Solaris, Linux i Tru64 obsługują konwersję z CCSID 897 do identyfikatorów CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 i 1252.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

290

Nie konwertuje na strony kodowe 932, 943, 954, 5050, 33722

IBM i

Strona kodowa:

290

Nie konwertuje na stronę kodową 932

AIX

Strona kodowa:

932

Nie konwertuje do stron kodowych 290, 897

5050

Nie konwertuje do stron kodowych 290, 897

33722

Nie konwertuje do stron kodowych 290, 897

HP-UX

Strona kodowa:

897

Nie konwertuje na strony kodowe 932, 943, 954, 5050, 33722

Linux

Strona kodowa:

943

Nie konwertuje do stron kodowych 290, 897

5050

Nie konwertuje do stron kodowych 290, 897

Solaris

Strona kodowa:

943

Nie konwertuje do stron kodowych 290, 897

5050

Nie konwertuje do stron kodowych 290, 897

HP Integrity NonStop Server

Strona kodowa:

943

Nie konwertuje do stron kodowych 290, 897

5050

Nie konwertuje do stron kodowych 290, 897

Japoński Kanji/Latin Mieszane

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka japońskiego Kanji/łacińskiego Mieszanego.

Poniższa tabela zawiera rodzime identyfikatory CCSID dla języka japońskiego Kanji/Latin Mieszane na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
IBM i, z/OS	1399, 5035 (patrz uwaga 1)
AIX	932, 5050, 33722 (patrz uwaga 2)
HP-UX	932, 954, 5039 (patrz uwaga 3)
Windows	932, 943 (patrz uwagi 4 i 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Uwaga:

1. 5035 to identyfikator CCSID związany ze stroną kodową 939
2. Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
3. Zestawy kodowe japan15 i SJIS w systemie HP-UX są reprezentowane przez CCSID 932. Mają one kilka znaków DBCS, które mają różne reprezentacje w SJIS, a więc 932 może zostać niepoprawnie przekształcone, jeśli konwersja nie jest wykonywana w systemie HP-UX. Produkt IBM MQ for HP-UX obsługuje 5039, poprawny identyfikator CCSID dla HP SJIS. Zmiana zbioru /var/mqm/conv/ccsid.tbl może zostać dokonana w celu zmiany identyfikatora CCSID używanego w zakresie od 932 do 5039.
4. Produkt Windows NT używa strony kodowej 932, ale najlepiej jest to przedstawić przy użyciu identyfikatora CCSID 943. Jednak nie wszystkie platformy produktu IBM MQ obsługują ten identyfikator CCSID.

W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można dokonać zmiany w pliku `./conv/table/ccsid.tbl`, która zmienia identyfikator CCSID używany na 943.

5. Produkt IBM MQ nie obsługuje stron kodowych w oparciu o standard JIS X 0213 (JIS2004).

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

1399

Nie konwertuje na strony kodowe 954, 5035, 5050, 33722

5035

Nie konwertuje na strony kodowe 954, 1399, 5050, 33722

IBM i

Strona kodowa:

1399

Nie konwertuje na stronę kodową 5039

5035

Nie konwertuje na stronę kodową 5039

HP-UX

Strona kodowa:

932

Nie konwertuje na strony kodowe 942, 943, 1399

954

Nie konwertuje na strony kodowe 942, 943, 1399

5039

Nie konwertuje na strony kodowe 942, 943, 1399

HP Integrity NonStop Server

Strona kodowa:

943

Nie konwertuje na stronę kodową 1399

5050

Nie konwertuje na stronę kodową 1399

Japoński Kanji/Katakana Mieszane

Szczegółowe informacje o identyfikatorach CCSID i konwersji CCSID dla języka japońskiego Kanji/Katakana Mieszanego.

Poniższa tabela przedstawia rodzime identyfikatory CCSID dla języka japońskiego Kanji/Katakana Mieszane na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
z/OS	1390, 5026 (patrz uwaga 1)
IBM i	5026 (zob. uwaga 1)
AIX	932, 5050, 33722 (patrz uwaga 2)
HP-UX	932, 954, 5039 (patrz uwaga 3)
Windows	932, 943 (patrz uwagi 4 i 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Uwaga:

- CCSID 1390 nie akceptuje małych liter. Wartość 5026 jest identyfikatorem CCSID związanym ze stroną kodową 930. CCSID 5026 jest identyfikatorem CCSID zgłoszonym w systemie IBM i po wybraniu opcji japońskiej firmy Katakana (DBCS).
- Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
- Zestawy kodowe japan15 i SJIS w systemie HP-UX są reprezentowane przez CCSID 932. Mają one kilka znaków DBCS, które mają różne reprezentacje w SJIS, a więc 932 może zostać niepoprawnie przekształcone, jeśli konwersja nie jest wykonywana w systemie HP-UX. Produkt IBM MQ for HP-UX obsługuje 5039, poprawny identyfikator CCSID dla HP SJIS. Zmiana zbioru /var/mqm/conv/ccsid.tbl może zostać dokonana w celu zmiany identyfikatora CCSID używanego w zakresie od 932 do 5039.

4. Produkt Windows NT używa strony kodowej 932, ale najlepiej jest to przedstawić przy użyciu identyfikatora CCSID 943. Jednak nie wszystkie platformy produktu IBM MQ obsługują ten identyfikator CCSID.

W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można dokonać zmiany w zbiorze `./conv/table/ccsid.tbl`, która zmienia identyfikator CCSID używany na 943.

5. Produkt IBM MQ nie obsługuje stron kodowych w oparciu o standard JIS X 0213 (JIS2004).

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

1390

Nie konwertuje na strony kodowe 954, 5026, 5050, 33722

Nie akceptuje małych liter.

5026

Nie konwertuje na strony kodowe 954, 1390, 5050, 33722

IBM i

Strona kodowa:

5026

Nie konwertuje na strony kodowe 1390, 5039

HP-UX

Strona kodowa:

932

Nie konwertuje na strony kodowe 942, 943, 1390

954

Nie konwertuje na strony kodowe 942, 943, 1390

5039

Nie konwertuje na strony kodowe 942, 943, 1390

HP Integrity NonStop Server

Strona kodowa:

943

Nie konwertuje na stronę kodową 1390

5050

Nie konwertuje na stronę kodową 1390

koreański

Szczegóły dotyczące konwersji CCSID i CCSID dla języka koreańskiego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka koreańskiego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
z/OS, IBM i	933, 1364
AIX, HP-UX, Linux, HP Integrity NonStop Server, Solaris	970

Platforma	Rodzime identyfikatory CCSID
Windows	949, 1363

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

933

Nie konwertuje na stronę kodową 970

1364

Nie konwertuje na stronę kodową 970

HP-UX

Strona kodowa:

970

Nie konwertuje na strony kodowe 949, 1363, 1364

chiński uproszczony

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka chińskiego uproszczonego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka chińskiego uproszczonego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
z/OS	935, 1388
IBM i	935, 1388
AIX	1383, 1386
HP-UX	1381 (zob. uwaga 1)
Windows	1381, 1386 (zob. uwaga 2)
Linux, HP Integrity NonStop Server, Solaris	1383

Uwaga:

1. Zestawy kodowe prc15 i hp15CN w systemie HP-UX są reprezentowane przez CCSID 1381.
2. Produkt Windows używa strony kodowej 936, ale najlepiej jest to przedstawić na podstawie identyfikatora CCSID 1386. Jednak nie wszystkie platformy produktu IBM MQ obsługują ten identyfikator CCSID.

W systemie IBM MQ for Windows CCSID 1381 jest używany do reprezentowania strony kodowej 936, ale można dokonać zmiany zbioru `.. /conv/table/ccsid.tbl`, która zmienia identyfikator CCSID używany na wartość 1386.

3. Produkt IBM MQ obsługuje fazę jednej z chińskich standardów GB18030.

W systemach z/OS, Linux, Windows i Solaris obsługa konwersji jest udostępniana między kodami Unicode (UTF-8 i UCS-2) i identyfikatorem CCSID 1388 (EBCDIC z rozszerzeniami GB18030), Unicode (UTF-8 i UCS-2) oraz identyfikatorem CCSID 5488 (pierwszy etap GB18030) oraz między identyfikatorem CCSID 1388 a identyfikatorem CCSID 5488.

Uwaga:

W systemie IBM isystem operacyjny obsługuje konwersję między kodowaniem Unicode (UTF-8 i UCS-2) a identyfikatorem CCSID 1388 (EBCDIC z rozszerzeniami GB18030).

W systemie HP-UX nie ma obecnie dostępnej obsługi w systemie operacyjnym HP11 dla systemu GB18030. W systemie HP11ipoprawka PHCO_26456 zapewnia obsługę konwersji między GB18030 (CCSID 5488) i Unicode. Obsługa konwersji między GB18030 i 1388 (EBCDIC) nie jest obsługiwana.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

935

Nie konwertuje na stronę kodową 1383

1388

Nie konwertuje na stronę kodową 1383

HP-UX

Strona kodowa:

1381

Nie konwertuje na strony kodowe 1383, 1386, 1388

chiński tradycyjny

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka chińskiego tradycyjnego.

W poniższej tabeli przedstawiono rodzime identyfikatory CCSID dla języka chińskiego tradycyjnego na obsługiwanych platformach:

Platforma	Rodzime identyfikatory CCSID
z/OS, IBM i	937
HP-UX	938, 950, 964 (patrz uwaga)
Windows	950
AIX, HP Integrity NonStop Server, Solaris, Linux	950, 964

Uwaga: Zestaw kodowy roc15 w systemie HP-UX jest reprezentowany przez CCSID 938.

Wszystkie platformy obsługują konwersję między swoimi własnymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

Strona kodowa:

937

Nie konwertuje na stronę kodową 964

1388

Nie konwertuje na stronę kodową 1383

HP-UX

Strona kodowa:

938

Nie konwertuje na stronę kodową 948

950

Nie konwertuje na stronę kodową 948

964

Nie konwertuje na stronę kodową 948

Linux, Solaris

Strona kodowa:

964

Nie konwertuje na stronę kodową 938

Obsługa konwersji produktu z/OS

Lista obsługiwanych konwersji CCSID.

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS	
CCSID	Konwertuje do i z CCSID
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 13121, 13488, 16804, 17248, 17584, 28709
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1250, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
278	37, 256, 273, 277, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
281	1047
282	500, 1047, 1200, 1208, 13488, 17584
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
293	1200, 1208, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 1097, 1098, 1114-1115, 1126, 1140-1149, 1200, 1208, 1252, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210-5211, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-924, 1004, 1009-1021, 1023, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097, 1100-1107, 1112, 1114-1115, 1122, 1124-1126, 1129-1133, 1137, 1140-1149, 1200, 1208, 1250-1258, 1275, 1280-1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5142, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9061, 13121, 13488, 16804, 17584, 28709
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1051, 1088-1089, 1097, 1098, 1112, 1114, 1122-1123, 1126, 1130, 1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25617, 25619, 25664, 28709
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25479, 25617, 25619, 25664, 28709
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709

<i>Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)</i>	
CCSID	Konwertuje do i z CCSID
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097, 1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1140, 1145-1146, 1148, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)

CCSID	Konwertuje do i z CCSID
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1148, 1149, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 9030, 12712, 13121, 13488, 16804, 17584, 28709
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869, 870, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5012, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 28709
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13488, 17584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584
1011	500, 1200, 1208, 13488, 17584
1012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 880, 897, 903, 912, 915-916, 920, 1009, 1026-1027, 1040-1043, 1051, 1088, 1112, 1122, 1131, 1200, 1208, 1251-1252, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5350, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1103	500
1104	500
1105	500
1106	500
1107	500
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200, 1208, 13488, 17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1097-1098, 1112, 1122-1123, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1251, 1254-1255, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13488, 17584, 28709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25473, 25617, 25619, 25664, 28709
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 25479, 25617, 25619, 25664, 28709
4933	837, 1200, 1208, 1380, 1385, 13488, 17584

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 17248, 25473, 25479, 25617, 25619, 28709
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097-1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 16804, 25473, 25479, 25617, 25619, 25664, 28709
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25479, 25617, 25619, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
8492	300-301, 941, 1351, 4396, 16684
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1153, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9238, 13121, 13488, 16804, 17248, 17584, 28709
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13488, 16804, 17248, 17584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13488, 17248, 17584, 25617, 25619, 25664, 28709
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 16684, 16804, 17248, 17584, 21427, 28709
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17584, 28709
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 21427, 28709
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 25479, 25617, 25619, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25617, 25619, 28709
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25619, 25664, 28709
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25617, 25664, 28709
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709

Tabela 121. Obsługa konwersji CCSID IBM MQ for z/OS (kontynuacja)	
CCSID	Konwertuje do i z CCSID
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

Obsługa konwersji produktu IBM i

Pełna lista identyfikatorów CCSID i konwersje obsługiwane przez produkt IBM można znaleźć w odpowiedniej publikacji produktu IBM i.

Obsługiwane strony kodowe są wymienione w sekcji [Obsługiwane odwzorowania CCSID](#).

Obsługa konwersji Unicode

Niektóre platformy obsługują konwersję danych użytkownika na kodowanie Unicode lub kodowanie Unicode. Obsługiwane są dwa formularze kodowania Unicode: UCS-2 (CCSID 1200, 13488 i 17584) oraz UTF-8 (CCSID 1208).

Termin *UCS-2* jest często używany zamiennie, ale jest niepoprawnie używany z produktem *UTF-16*. *UCS-2* jest kodowaniem o stałej szerokości, w którym każdy znak zajmuje 2 bajty. *UTF-16* to kodowanie o zmiennej szerokości, które jest superzestawem *UCS-2*. Oprócz dwubajtowych znaków *UCS-2*, *UTF-16* zawiera znaki, znane jako pary odpowiedników, które mają długość 4 bajtów. Produkt IBM MQ nie obsługuje par zastępców. Obsługa znaków *UTF-16* i *UTF-8* w produkcie IBM MQ jest więc ograniczona do tych znaków Unicode, które mogą być kodowane w *UCS-2*.

Uwaga: Produkt IBM MQ nie obsługuje identyfikatorów CCSID menedżera kolejek *UCS-2*, dlatego dane nagłówek komunikatu nie mogą być zakodowane w *UCS-2*.

Obsługa systemu IBM MQ AIX dla kodu Unicode

W poniższej tabeli dla identyfikatorów CCSID obsługiwane są identyfikatory CCSID w systemie IBM MQ for AIX do i z identyfikatorów CCSID Unicode.

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

Obsługa systemu IBM MQ HP-UX dla kodu Unicode

W przypadku IBM MQ for HP-UX konwersji do i z CCSID Unicode obsługiwane są identyfikatory CCSID wymienione w poniższej tabeli.

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		

Obsługa produktów IBM MQ for Windows, Solaris Linux dla kodu Unicode

W następujących tabelach: IBM MQ for Windows, IBM MQ for Solaris IBM MQ dla konwersji Linux oraz z CCSID Unicode są obsługiwane dla identyfikatorów CCSID w poniższej tabeli.

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)		

Uwagi:

1. 931 używa 939 do konwersji.
2. 932 wykorzystuje 942 do konwersji.
3. 938 wykorzystuje 948 do konwersji.
4. 954 i 33722 używają 5050 do konwersji.
5. Tylko w systemach Windows, Linux Solaris .

Obsługa produktu IBM i dla kodu Unicode

Szczegółowe informacje na temat obsługi UNICODE zawiera odpowiednia publikacja IBM i odnosząca się do używanego systemu operacyjnego.

Obsługa produktu IBM MQ for z/OS dla kodu Unicode

W przypadku IBM MQ for z/OS konwersji do i z CCSID Unicode obsługiwane są następujące identyfikatory CCSID:

37	256	259	273	275	277
278	280	282	284	285	290
293	297	300	301	367	420
423	424	437	500	720	737
775	803	806	808	813	819
833	834	835	836	837	838
848	849	850	851	852	855
856	857	858	859	860	861
862	863	864	865	866	867
868	869	870	871	872	874
875	878	880	891	895	896
897	901	902	903	904	905
912	914	915	916	918	920
921	922	923	924	927	928
930	932	933	935	937	939
941	942	943	944	946	947
948	949	950	951	1004	1006
1008	1009	1010	1011	1012	1013
1014	1015	1016	1017	1018	1019
1025	1026	1027	1040	1041	1042
1043	1046	1047	1051	1088	1089
1097	1098	1112	1114	1115	1122
1123	1124	1125	1126	1129	1130
1131	1132	1133	1137	1140	1141
1142	1143	1144	1145	1146	1147
1148	1149	1153	1154	1155	1156
1157	1158	1159	1160	1161	1162
1164	1200	1208	1250	1251	1252
1253	1254	1255	1256	1257	1258
1275	1276	1277	1280	1281	1282
1283	1284	1285	1351	1362	1363
1364	1370	1371	1380	1381	1385

1386	1388	1390	1399	4899	4909
4930	4933	4948	4951	4952	4960
4971	5012	5039	5104	5123	5142
5210	5346	5347	5348	5349	5350
5351	5352	5353	5354	5488	8482
8612	9027	9030	9044	9048	9049
9056	9061	9066	9238	9449	1166
12712	13121	13218	13488	16684	16804
17248	17584	21427	28709		

Standardy kodowania na platformach 64-bitowych

Ta sekcja zawiera informacje na temat kodowania standardów na platformach 64-bitowych i preferowanych typach danych.

Preferowane typy danych

Te typy nigdy nie zmieniają wielkości i są dostępne zarówno na 32-bitowych, jak i 64-bitowych platformach IBM MQ :

Nazwa	Długość
MQLONG	4 bajty
MQULONG	4 bajty
MQINT32	4 bajty
MQUINT32	4 bajty
MQINT64	8 bajtów
MQUINT64	8 bajtów

Standardowe typy danych

Informacje na temat standardowych typów danych w 32-bitowych aplikacjach UNIX, 64-bitowych UNIX i 64-bitowych Windows .

32-bitowe aplikacje produktu UNIX

Ta sekcja jest uwzględniana w celu porównania i jest oparta na produkcie Solaris. Wszystkie różnice z innymi Platformy UNIX są zauważone:

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty
long	4 bajty
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów

Nazwa	Długość
długie podwójne	16 bajtów
	Należy pamiętać, że w systemach AIX i Linux PPC długa liczba znaków dwubajtowych wynosi 8 bajtów.
wskaźnik	4 bajty
ptrdiff_t	4 bajty
wielkość_t	4 bajty
time_t	4 bajty
clock_t	4 bajty
wchar_t	4 bajty
	Należy zauważyć, że w systemie AIX wartość wchar_t wynosi 2 bajty.

64-bitowe aplikacje produktu UNIX

Ta sekcja jest oparta na produkcie Solaris. Wszystkie różnice z innymi Platformy UNIX są zauważone:

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty
long	8 bajtów
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów
długie podwójne	16 bajtów
	Należy pamiętać, że w systemach AIX i Linux PPC długa liczba znaków dwubajtowych wynosi 8 bajtów.
wskaźnik	8 bajtów
ptrdiff_t	8 bajtów
wielkość_t	8 bajtów
time_t	8 bajtów
clock_t	8 bajtów
	Należy zauważyć, że w drugim Platformy UNIX zegark_t jest 4 bajtów.
wchar_t	4 bajty
	Należy zauważyć, że w systemie AIX wartość wchar_t wynosi 2 bajty.

Aplikacje 64-bitowe produktu Windows

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty

Nazwa	Długość
long	4 bajty
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów
długie podwójne	8 bajtów
wskaźnik	8 bajtów
Należy zauważyć, że wszystkie wskaźniki to 8 bajtów.	
ptrdiff_t	8 bajtów
wielkość_t	8 bajtów
time_t	8 bajtów
clock_t	4 bajty
wchar_t	2 bajty
Word	2 bajty
DWORD	4 bajty
aplikacji	8 bajtów
HFILE	4 bajty

Uwagi dotyczące kodowania w systemie Windows

HANDLE hf;

Użyj

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

Nie używaj

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

ponieważ powoduje to wystąpienie błędu.

wielkość_t fgets len

Użyj

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

Nie używaj

```
int len;

while (fgets(string1, len, fp) != NULL)
    len = strlen(buffer);
```

printf

Użyj

```
printf("My struct pointer: %p", pMyStruct);
```

Nie używaj

```
printf("My struct pointer: %x", pMyStruct);
```

Jeśli potrzebne są dane wyjściowe w postaci szesnastkowej, należy wydrukować górną i dolną 4 bajty oddzielnie.

char * ptr

Użyj

```
char * ptr1;
char * ptr2;
size_t buflen;

buflen = ptr2 - ptr1;
```

Nie używaj

```
char *ptr1;
char *ptr2;
UINT32 buflen;

buflen = ptr2 - ptr1;
```

alignBytes

Użyj

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

Nie używaj

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

Użyj

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

Nie używaj

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

Użyj

```
MQLONG SBCSprt;  
sscanf(line, "%d", &SBCSprt);
```

Nie używaj

```
MQLONG SBCSprt;  
sscanf(line, "%1d", &SBCSprt);
```

Program %1d próbuje umieścić typ 8-bajtowy w 4-bajtowym typie, a w przypadku rzeczywistego typu danych programu long należy używać tylko %1 . Wartości MQLONG, UINT32 i INT32 są zdefiniowane jako cztery bajty, takie same jak int na wszystkich platformach IBM MQ :



Skorowidz programistyczny aplikacji IBM i (ILE/RPG)

Programowanie aplikacji dla produktu IBM i

Następujące tematy ułatwiają tworzenie aplikacji dla produktu IBM i:

- [“Opisy typów danych” na stronie 1009](#)
- [“Wywołania funkcji” na stronie 1270](#)
- [“Atrybuty obiektów” na stronie 1395](#)
- [“Aplikacje” na stronie 1441](#)
- [“Kody powrotu dla IBM i \(ILE RPG\)” na stronie 1455](#)
- [“Reguły sprawdzania poprawności opcji MQI dla produktu IBM i \(ILE RPG\)” na stronie 1456](#)
- [“Kodowanie komputera w systemie IBM i” na stronie 1459](#)
- [“Opcje raportów i flagi komunikatów” na stronie 1462](#)

Tryb zgodności dla aplikacji RPG w systemie IBM i

Ten tryb działania był wymagany dla aplikacji, które są zapisywane przed wersją MQSeries 5.1. Kolejne wersje tego produktu udostępniają kompatybilne środowisko wykonawcze dla tych aplikacji, mimo że struktury copybook potrzebne do kompilowania ich zostały usunięte w produkcie IBM WebSphere MQ 6.0. Mimo że połączenie dynamiczne (tryb zgodności) jest udostępniane w bibliotece QMQM dla wersji produktu po MQSeries w wersji 5.1, aplikacje, które używają tego trybu zgodności, muszą zostać zrekompilowane w celu użycia statycznych wywołań MQ udostępnianych przez programy usługowe LIBMQM i LIBMQM_R.

Informacje pokrewne

[Projektowanie aplikacji](#)

Opisy typów danych

Ta kolekcja tematów zawiera opisy typów danych używanych w programowaniu produktu IBM i .

Konwencje używane w opisie typów danych

Dla każdego elementarnego typu danych informacje te dają opis jego użycia w postaci, która jest niezależna od języka programowania. Następuje to po typowych deklaracjach w wersji ILE języka programowania RPG. Definicje elementarnych typów danych są zawarte w tym miejscu, aby zapewnić spójność. W języku RPG używane są specyfikacje D, w których pola robocze mogą być deklarowane przy użyciu dowolnych atrybutów. Można to jednak zrobić w specyfikacjach obliczeniowych, w których pole jest używane.

Aby korzystać z elementarnych typów danych, należy utworzyć następujące elementy:

- Element /COPY zawierający wszystkie typy danych, lub
- Zewnętrzna struktura danych (PF) zawierająca wszystkie typy danych. Następnie należy określić pola robocze zawierające atrybuty "LIKE" odpowiednie pole typu danych.

Korzyściami z drugiej opcji jest to, że definicje mogą być używane jako 'FIELD REFERENCE FILE' dla innych obiektów IBM i . Jeśli definicja typu danych IBM MQ ulegnie zmianie, jest to stosunkowo prosta sprawa do ponownego utworzenia tych obiektów.

Elementarne typy danych

Wszystkie inne typy danych opisane w tej sekcji równają się bezpośrednio do tych podstawowych typów danych lub do agregowania tych podstawowych typów danych (tablic lub struktur).

Tabela 122. Elementarne typy danych	
Typ danych	Reprezentacja
MQBOOL	10-cyfrowa liczba całkowita ze znakiem
MQBYTE	1-bajtowe pole alfanumeryczne
MQBYTE16	16-bajtowe pole alfanumeryczne
MQBYTE24	24-bajtowe pole alfanumeryczne
MQBYTE32	32-bitowe pole alfanumeryczne
MQBYTE64	64-bitowe pole alfanumeryczne
MQCHAR	1-bajtowe pole alfanumeryczne
MQCHAR4	4-bajtowe pole alfanumeryczne
MQCHAR8	8-bajtowe pole alfanumeryczne
MQCHAR12	12-bajtowe pole alfanumeryczne
MQCHAR16	16-bajtowe pole alfanumeryczne
MQCHAR20	20-bajtowe pole alfanumeryczne
MQCHAR28	28-bajtowe pole alfanumeryczne
MQCHAR32	32-bitowe pole alfanumeryczne
MQCHAR48	48-bajtowe pole alfanumeryczne
MQCHAR64	64-bitowe pole alfanumeryczne
MQCHAR128	128-bajtowe pole alfanumeryczne
MQCHAR256	256-bajtowe pole alfanumeryczne
MQFLOAT32	4-bajtowa liczba zmiennoprzecinkowa
MQFLOAT64	8-bajtowa liczba zmiennoprzecinkowa
MQHCONFIG	Uchwyt konfiguracji
MQHCONN	10-cyfrowa liczba całkowita ze znakiem
MQHMSG	Uchwyt komunikatu, który zapewnia dostęp do komunikatu
MQHOBJ	10-cyfrowa liczba całkowita ze znakiem
MQINT8	8-bitowa liczba całkowita ze znakiem

Tabela 122. Elementarne typy danych (kontynuacja)

Typ danych	Reprezentacja
MQINT16	16-bitowa liczba całkowita ze znakiem
MQINT32	32-bitowa liczba całkowita ze znakiem
MQINT64	64-bitowa liczba całkowita ze znakiem
MQLONG	32-bitowa liczba całkowita ze znakiem
MQPID	Identyfikator procesu
MQPTR	Wskaźnik
Identyfikator MQTID	Identyfikator wątku
MQUINT8	8-bitowa liczba całkowita bez znaku
MQUINT16	16-bitowa liczba całkowita bez znaku
MQUINT32	32-bitowa liczba całkowita bez znaku
MQUINT64	64-bitowa liczba całkowita bez znaku
MQULONG	32-bitowa liczba całkowita bez znaku
PMQACH	Wskaźnik do struktury danych typu MQACH
PMQAIR	Wskaźnik do struktury danych typu MQAIR
PMQAXC	Wskaźnik do struktury danych typu MQAXC
PMAXP	Wskaźnik do struktury danych typu MAXP
PMQBMHO	Wskaźnik do struktury danych typu MQBMHO
PMQBO	Wskaźnik do struktury danych typu MQBO
PMQBOOL	Wskaźnik do danych typu MQBOOL
PMQBYTE	Wskaźnik do danych typu MQBYTE
PMQBYTEN	Wskaźnik do danych typu MQBYTEN
PMQCBC	Wskaźnik do struktury danych typu MQCBC
PMQCBD	Wskaźnik do struktury danych typu MQCBD
PMQCHAR	Wskaźnik do struktury danych typu MQCHAR
PMQCHARV	Wskaźnik do struktury danych typu MQCHARV
PMQCHARN	Wskaźnik do danych typu MQCHARn
PMQCIH	Wskaźnik do struktury danych typu MQCIH
PMQCMHO	Wskaźnik do struktury danych typu MQCMHO
PMQCNO	Wskaźnik do struktury danych typu MQCNO
PMQCSP	Wskaźnik do struktury danych typu MQCSP
PMQCTLO	Wskaźnik do struktury danych typu MQCTLO
PMQDH	Wskaźnik do struktury danych typu MQDH
PMQDHO	Wskaźnik do struktury danych typu MQDHO
PMQDLH	Wskaźnik do struktury danych typu MQDLH

Tabela 122. Elementarne typy danych (kontynuacja)

Typ danych	Reprezentacja
PMQDMHO	Wskaźnik do struktury danych typu MQDMHO
PMQDMPO	Wskaźnik do struktury danych typu MQDMPO
PMQEPH	Wskaźnik do struktury danych typu MQEPH
PMQFLOAT32	Wskaźnik do danych typu MQFLOAT32
PMQFLOAT64	Wskaźnik do danych typu MQFLOAT64
PMQFUNC	Wskaźnik do funkcji
PMQGM0	Wskaźnik do struktury danych typu MQGM0
PMQHCONFIG	Wskaźnik do danych typu MQHCONFIG
PMQHCONN	Wskaźnik do danych typu MQHCONN
PMQHMSG	Wskaźnik do danych typu MQHMSG
PMQHOBj	Wskaźnik do danych typu MQHOBj
PMQIIH	Wskaźnik do struktury danych typu MQIIH
PMQIMPO	Wskaźnik do struktury danych typu MQIMPO
PMQINT8	Wskaźnik do danych typu MQINT8
PMQINT16	Wskaźnik do danych typu MQINT16
PMQINT32	Wskaźnik do danych typu MQINT32
PMQINT64	Wskaźnik do danych typu MQINT64
PMQLONG	Wskaźnik do danych typu MQLONG
PMQMD	Wskaźnik do struktury danych typu MQMD
PMQMDE	Wskaźnik do struktury danych typu MQMDE
PMQMD1	Wskaźnik do struktury danych typu MQMD1
PMQMD2	Wskaźnik do struktury danych typu MQMD2
PMQMHBO	Wskaźnik do struktury danych typu MQMHBO
PMQOD	Wskaźnik do struktury danych typu MQOD
PMQOR	Wskaźnik do struktury danych typu MQOR
PMQPD	Wskaźnik do struktury danych typu MQPD
PMQPID	Wskaźnik do identyfikatora procesu MQPID
PMQPMO	Wskaźnik do struktury danych typu MQPMO
PMQPTR	Wskaźnik do danych typu MQPTR
PMQRFH	Wskaźnik do struktury danych typu MQRFH
PMQRFH2	Wskaźnik do struktury danych typu MQRFH2
PMQRMH	Wskaźnik do struktury danych typu MQRMH
PMQRR	Wskaźnik do struktury danych typu MQRR
PMQSCO	Wskaźnik do struktury danych typu MQSCO

Tabela 122. Elementarne typy danych (kontynuacja)

Typ danych	Reprezentacja
PMQSD	Wskaźnik do struktury danych typu MQSD
PMQSMPO	Wskaźnik do struktury danych typu MQSMPO
PMQSRO	Wskaźnik do struktury danych typu MQSRO
PMQSTS	Wskaźnik do struktury danych typu MQSTS
Identyfikator PMQTID	Wskaźnik do identyfikatora wątku MQTID
PMQTM	Wskaźnik do struktury danych typu MQTM
PMQTMCM2	Wskaźnik do struktury danych typu MQTMCM2
PMQUINT8	Wskaźnik do danych typu MQUINT8
PMQUINT16	Wskaźnik do danych typu MQUINT16
PMQUINT32	Wskaźnik do danych typu MQUINT32
PMQUINT64	Wskaźnik do danych typu MQUINT64
PMQULONG	Wskaźnik do danych typu MQULONG
PMQVOID	Wskaźnik
PMQWIH	Wskaźnik do struktury danych typu MQWIH
PMQXQH	Wskaźnik do struktury danych typu MQXQH

MQBOOL

Typ danych MQBOOL reprezentuje wartość boolową. Wartość 0 oznacza wartość false. Każda inna wartość reprezentuje wartość true.

Obiekt MQBOOL musi być wyrównany w taki sposób, aby był zgodny z typem danych MQLONG.

MQBYTE-Bajt

Typ danych MQBYTE reprezentuje jeden bajt danych.

Żadna konkretna interpretacja nie jest umieszczana na bajcie-jest traktowana jako ciąg bitów, a nie jako binarny numer czy znak. Specjalne wyrównanie nie jest wymagane.

Tablica zmaterializowana MQBYTE jest czasami używana do reprezentowania obszaru pamięci głównej z naturą, która nie jest znana menedżerowi kolejek. Na przykład obszar może zawierać dane komunikatu aplikacji lub strukturę. Wyrównanie graniczne tego obszaru musi być zgodne z charakterem zawartych w nim danych.

MQBYTEN-łańcuch n bajtów

Każdy typ danych MQBYTEN reprezentuje łańcuch n bajtów.

Gdzie n może przyjmować jedną z następujących wartości:

- 16, 24, 32 lub 64.

Każdy bajt jest opisany przez typ danych MQBYTE. Specjalne wyrównanie nie jest wymagane.

Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełnione wartościami pustymi w celu wypełnienia łańcucha.

Gdy menedżer kolejek zwraca łańcuchy bajtowe do aplikacji (na przykład w wywołaniu MQGET), menedżer kolejek zawsze kłocka z wartościami pustymi do zdefiniowanej długości łańcucha.

Dostępne są stałe, które definiują długości pól łańcucha bajtów.

MQCHAR-znak

Typ danych MQCHAR reprezentuje pojedynczy znak.

Identyfikator kodowanego zestawu znaków tego znaku jest taki sam, jak w przypadku menedżera kolejek (patrz atrybut *CodedCharSetId* w temacie *CodedCharSetId*). Specjalne wyrównanie nie jest wymagane.

Uwaga: Dane komunikatu aplikacji określone w wywołaniach MQGET, MQPUT i MQPUT1 są opisane w typie danych MQBYTE, a nie w typie danych MQCHAR.

MQCHARn-łańcuch n znaków

Każdy typ danych MQCHARn reprezentuje łańcuch znaków *n* znaków.

Gdzie *n* może przyjmować jedną z następujących wartości:

- 4, 8, 12, 16, 20, 28, 32, 48, 64, 128 lub 256

Każdy znak jest opisany przez typ danych MQCHAR. Specjalne wyrównanie nie jest wymagane.

Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełniane spacjami, aby wypełnić łańcuch. W niektórych przypadkach znak o kodzie zero może być używany do przedwczesnego zakończenia łańcucha, zamiast dopełniania odstępami; znak o kodzie zero i znaki następujące po nim są traktowane jako odstępy, aż do długości określonej długości łańcucha. Miejsca, w których można użyć wartości NULL, są identyfikowane w opisach wywołania i typu danych.

Gdy menedżer kolejek zwraca łańcuchy znaków do aplikacji (na przykład w wywołaniu MQGET), menedżer kolejek zawsze podkłada odstępy do zdefiniowanej długości łańcucha; menedżer kolejek nie używa znaku o kodzie zero do odkodowania łańcucha.

Dostępne są stałe, które definiują długości pól łańcucha znaków.

MQFLOAT32

Typ danych MQFLOAT32 jest 32-bitową liczbą zmiennopozycyjną reprezentowaną przy użyciu standardowego formatu zmiennopozycyjnego IEEE.

Wartość MQFLOAT32 musi być wyrównana w 4-bajtowej granicy.

MQFLOAT64

Typ danych MQFLOAT64 jest 64-bitową liczbą zmiennopozycyjną reprezentowaną przy użyciu standardowego formatu zmiennopozycyjnego IEEE.

Wartość MQFLOAT64 musi być wyrównana w 8-bajtowej granicy.

MQHCONFIG-uchwyt konfiguracji

Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla określonej usługi instalowalnej. Uchwyt konfiguracji musi być wyrównany względem jego naturalnej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko w celu zapewnienia równości.

MQHCONN-uchwyt połączenia

Typ danych MQHCONN reprezentuje uchwyt połączenia, tj. połączenie z określonym menedżerem kolejek.

Uchwyt połączenia musi być wyrównany względem jego naturalnej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko w celu zapewnienia równości.

MQHMSG-uchwyt komunikatu

Typ danych MQHMSG reprezentuje uchwyt komunikatu, który zapewnia dostęp do komunikatu.

Uchwyt komunikatu musi być wyrównany na 8-bajtowej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko w celu zapewnienia równości.

MQHOBJ-Uchwyt obiektu

Typ danych MQHOBJ reprezentuje uchwyt obiektu, który daje dostęp do obiektu.

Uchwyt obiektu musi być wyrównany względem jego naturalnej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko w celu zapewnienia równości.

MQINT8 -8-bitowa liczba całkowita ze znakiem

Typ danych MQINT8 jest 8-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -128 do +127, chyba że kontekst został ograniczony przez kontekst.

MQINT16 -16-bitowa liczba całkowita ze znakiem

Typ danych MQINT16 jest 16-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -32 768 do +32 767, chyba że kontekst został ograniczony przez kontekst.

Wartość MQINT16 musi być wyrównana do granicy dwubajtowej.

MQINT32 -32-bitowa liczba całkowita

Typ danych MQINT32 jest 32-bitową liczbą całkowitą ze znakiem.

Jest on równoważny z MQLONG.

MQINT64 -liczba całkowita 64-bitowa

Typ danych MQINT64 jest 64-bitową liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, o ile nie jest to inaczej ograniczone przez kontekst.

W przypadku języka COBOL poprawny zakres jest ograniczony do -999 999 999 999 999 do +999 999 999 999 999. Wartość MQINT64 powinna być wyrównana w 8-bajtowej granicy.

MQLONG-długa liczba całkowita

Typ danych MQLONG jest 32-bitową binarną liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, chyba że kontekst nie jest ograniczony przez kontekst, wyrównany względem jego naturalnej granicy.

MQPID-identyfikator procesu

Identyfikator procesu IBM MQ .

Jest to ten sam identyfikator używany w śledzeniu IBM MQ i zrzutach FFST , ale może być inny niż identyfikator procesu systemu operacyjnego.

MQPTR-wskaźnik

Typ danych MQPTR to adres danych dowolnego typu. Wskaźnik musi być wyrównany na jego granicy naturalnej; jest to 16-bajtowa granica na IBM i.

Niektóre języki programowania obsługują wskaźniki o typie strukturalnego. MQI używa ich również w kilku przypadkach.

MQTID-identyfikator wątku

Identyfikator wątku MQ .

Jest to ten sam identyfikator używany w danych śledzenia MQ i zrzutach FFST , ale może być inny niż identyfikator wątku systemu operacyjnego.

MQUINT8 -8-bitowa liczba całkowita bez znaku

Typ danych MQUINT8 jest 8-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +255, chyba że kontekst jest ograniczony przez kontekst.

MQUINT16 -16-bitowa liczba całkowita bez znaku

Typ danych MQUINT16 jest 16-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +65 535, chyba że kontekst jest ograniczony przez kontekst.

Wartość MQUINT16 musi być wyrównana do granicy dwubajtowej.

MQUINT32 -32-bitowa liczba całkowita bez znaku

Typ danych MQUINT32 jest 32-bitową liczbą całkowitą bez znaku. Jest to odpowiednik MQULONG.

MQUINT64 -liczba całkowita z 64-bitową liczbą bez znaku

Typ danych MQUINT64 jest 64-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +18 446 744 073 709 551 615, chyba że kontekst został ograniczony przez kontekst.

W przypadku języka COBOL, poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999 999 999 999. Wartość MQUINT64 powinna być wyrównana w 8-bajtowej granicy.

MQULONG-32-bitowa liczba całkowita bez znaku

Typ danych MQULONG jest 32-bitową, niepodpisaną binarną liczbą całkowitą, która może przyjmować dowolną wartość z zakresu od 0 do + 4 294 967 294, chyba że kontekst został ograniczony przez kontekst.

Wartość MQULONG musi być wyrównana na granicy 4-bajtowej.

PMQACH-wskaźnik do struktury danych typu MQACH

Wskaźnik do struktury danych typu MQACH.

PMQAIR-wskaźnik do struktury danych typu MQAIR

Wskaźnik do struktury danych typu MQAIR.

PMQAXC-wskaźnik do struktury danych typu MQAXC

Wskaźnik do struktury danych typu MQAXC.

PMQAXP-wskaźnik do struktury danych typu MQAXP

Wskaźnik do struktury danych typu MQAXP.

PMQBMHO-wskaźnik do struktury danych typu MQBMHO

Wskaźnik do struktury danych typu MQBMHO.

PMQBO-wskaźnik do struktury danych typu MQBO

Wskaźnik do struktury danych typu MQBO.

PMQBOOL-wskaźnik do danych typu MQBOOL

Wskaźnik do danych typu MQBOOL.

Wskaźnik do danych typu MQBOOL.

PMQBYTE-wskaźnik do typu danych MQBYTE

Wskaźnik do typu danych MQBYTE.

PMQBYTEN-wskaźnik do struktury danych typu MQBYTEN

Wskaźnik do struktury danych typu MQBYTEN, gdzie n może mieć wartość 8, 12, 16, 24, 32, 40, 48 lub 128.

PMQCBC-wskaźnik do struktury danych typu MQCBC

Wskaźnik do struktury danych typu MQCBC.

PMQCBD-wskaźnik do struktury danych typu MQCBD

Wskaźnik do struktury danych typu MQCBD.

PMQCHAR-wskaźnik do danych typu MQCHAR

Wskaźnik do danych typu MQCHAR.

PMQCHARV-wskaźnik do struktury danych typu MQCHARV

Wskaźnik do struktury danych typu MQCHARV.

PMQCHARn-wskaźnik do typu danych MQCHARn

Wskaźnik do typu danych MQCHARn, gdzie n może mieć wartość 4, 8, 12, 20, 28, 32, 64, 128, 256, 264.

PMQCIH-wskaźnik do struktury danych typu MQCIH

Wskaźnik do struktury danych typu MQCIH.

PMQCMHO-wskaźnik do struktury danych typu MQCMHO

Wskaźnik do struktury danych typu MQCMHO.

PMQCNO-wskaźnik do struktury danych typu MQCNO

Wskaźnik do struktury danych typu MQCNO.

PMQCSP-wskaźnik do struktury danych typu MQCSP

Wskaźnik do struktury danych typu MQCSP.

PMQCTLO-wskaźnik do struktury danych typu MQCTLO

Wskaźnik do struktury danych typu MQCTLO.

PMQDH-wskaźnik do struktury danych typu MQDH

Wskaźnik do struktury danych typu MQDH.

PMQDHO-wskaźnik do struktury danych typu MQDHO

Wskaźnik do struktury danych typu MQDHO.

PMQDLH-wskaźnik do struktury danych typu MQDLH

Wskaźnik do struktury danych typu MQDLH.

PMQDMHO-wskaźnik do struktury danych typu MQDMHO

Wskaźnik do struktury danych typu MQDMHO.

PMQDMPO-wskaźnik do struktury danych typu MQDMPO

Wskaźnik do struktury danych typu MQDMPO.

Wskaźnik do struktury danych typu MQDMPO.

PMQEPH-wskaźnik do struktury danych typu MQEPH

Wskaźnik do struktury danych typu MQEPH.

PMQFLOAT32 -wskaźnik do danych typu MQFLOAT32

Wskaźnik do danych typu MQFLOAT32.

PMQFLOAT64 -wskaźnik do danych typu MQFLOAT64

Wskaźnik do danych typu MQFLOAT64.

PMQFUNC-wskaźnik do funkcji

Wskaźnik do funkcji.

PMQGMO-wskaźnik do struktury danych typu MQGMO

Wskaźnik do struktury danych typu MQGMO.

PMQHCONFIG-wskaźnik do typu danych MQHCONFIG

Wskaźnik do typu danych MQHCONFIG.

PMQHCONN-wskaźnik do typu danych MQHCONN

Wskaźnik do typu danych MQHCONN.

PMQHMSG-wskaźnik do typu danych MQHMSG

Wskaźnik do typu danych MQHMSG.

PMQHOBJ-wskaźnik do danych typu MQHOBJ

Wskaźnik do danych typu MQSMPO.

PMQIIH-wskaźnik do struktury danych typu MQIIH

Wskaźnik do struktury danych typu MQIIH.

PMQIMPO-wskaźnik do struktury danych typu MQIMPO

Wskaźnik do struktury danych typu MQIMPO.

PMQINT8 -wskaźnik do danych typu MQINT8

Wskaźnik do danych typu MQINT8.

PMQINT16 -wskaźnik do danych typu MQINT16

Wskaźnik do danych typu MQINT16.

PMQINT32 -Pointer do danych typu MQINT32

Typ danych PMQINT32 jest wskaźnikiem do danych typu MQINT32. Jest on równoważny z PMQLONG.

PMQINT64 -Pointer do danych typu MQINT64

Typ danych PMQINT64 jest wskaźnikiem do danych typu MQINT64.

PMQLONG-wskaźnik do danych typu MQLONG

Wskaźnik do danych typu MQLONG.

PMQMD-wskaźnik do struktury typu MQMD

Wskaźnik do struktury typu MQMD.

PMQMDE-wskaźnik do struktury danych typu MQMDE

Wskaźnik do struktury danych typu MQMDE.

PMQMDEI-wskaźnik do struktury danych typu MQMDI

Wskaźnik do struktury danych typu MQMDI.

PMQMD2 -wskaźnik do struktury danych typu MQMD2

Wskaźnik do struktury danych typu MQMD2

PMQMHBO-wskaźnik do struktury danych typu MQMHBO

Wskaźnik do struktury danych typu MQMHBO.

PMQOD-wskaźnik do struktury danych typu MQOD

Wskaźnik do struktury danych typu MQOD.

PMQOR-wskaźnik do struktury danych typu MQOR

Wskaźnik do struktury danych typu MQOR.

PMQPD-wskaźnik do struktury danych typu MQPD

Wskaźnik do struktury danych typu MQPD.

PMQPID-wskaźnik do identyfikatora procesu

Wskaźnik do identyfikatora procesu.

PMQPMO-wskaźnik do struktury danych typu MQPMO

Wskaźnik do struktury danych typu MQPMO.

PMQPTR-wskaźnik do danych typu MQPTR

Wskaźnik do danych typu MQPTR.

PMQRFH-wskaźnik do struktury danych typu MQRFH

Wskaźnik do struktury danych typu MQRFH.

PMQRFH2 -wskaźnik do struktury danych typu MQRFH2

Wskaźnik do struktury danych typu MQRFH2.

.

PMQRMH-wskaźnik do struktury danych typu MQRMH

Wskaźnik do struktury danych typu MQRMH.

PMQRR-wskaźnik do struktury danych typu MQRR

Wskaźnik do struktury danych typu MQRR.

PMQSCO-wskaźnik do struktury danych typu MQSCO

Wskaźnik do struktury danych typu MQSCO.

.

PMQSD-wskaźnik do struktury danych typu MQSD

Wskaźnik do struktury danych typu MQSD.

PMQSMPO-wskaźnik do struktury danych typu MQSMPO

Wskaźnik do struktury danych typu MQSMPO.

PMQSRO-wskaźnik do struktury danych typu MQSRO

Wskaźnik do struktury danych typu MQSRO.

PMQSTS-wskaźnik do struktury danych typu MQSTS

Wskaźnik do struktury danych typu MQSTS.

PMQTID-wskaźnik do struktury danych typu MQTID

Wskaźnik do struktury danych typu MQTID.

PMQTM-wskaźnik do struktury danych typu MQTM

Wskaźnik do struktury danych typu MQTM.

PMQPMC2 -wskaźnik do struktury danych typu MQPMC2

Wskaźnik do struktury danych typu MQPMC2.

PMQUINT8 -wskaźnik do danych typu MQUINT8

Wskaźnik do danych typu MQUINT8.

PMQUINT16 -wskaźnik do danych typu MQUINT16

Wskaźnik do danych typu MQUINT16.

PMQUINT32 -Pointer do danych typu MQUINT32

Typ danych PMQUINT32 jest wskaźnikiem do danych typu MQUINT32. Jest on równoważny z PMQULONG.

PMQUINT64 -Pointer do danych typu MQUINT64

Typ danych PMQUINT64 jest wskaźnikiem do danych typu MQUINT64.

PMQULONG-wskaźnik do danych typu MQULONG

Wskaźnik do danych typu MQULONG.

PMQVOID-wskaźnik

Wskaźnik.

PMQWIH-wskaźnik do struktury danych typu MQWIH

Wskaźnik do struktury danych typu MQWIH.

PMQXQH-wskaźnik do struktury danych typu MQXQH

Wskaźnik do struktury danych typu MQXQH.

Uwagi dotyczące języka

Ten temat zawiera informacje pomocne podczas korzystania z interfejsu MQI z języka programowania RPG.

Niektóre z tych zagadnień językowych są następujące:

- [“Kopiuj pliki” na stronie 1022](#)
- [“Wywołania” na stronie 1024](#)
- [“Parametry wywołania” na stronie 1024](#)
- [“Struktury” na stronie 1024](#)
- [“Stałe nazwane” na stronie 1025](#)
- [“Procedury MQI” na stronie 1025](#)
- [“Zagadnienia związane z wątkami” na stronie 1025](#)
- [“Kontrola transakcji” na stronie 1026](#)
- [“Kodowanie połączeń powiązanych” na stronie 1026](#)
- [“Konwencje z adnotacjami” na stronie 1027](#)

Kopiuj pliki

Udostępniane są różne pliki COPY, które ułatwiają pisanie programów aplikacji RPG, które korzystają z kolejkovania komunikatów. Istnieją trzy zestawy plików COPY:

- Pliki COPY o nazwach kończących się literą *G* są przeznaczone do użytku z programami, które używają łączenia statycznego. Te pliki są inicjowane z wyjątkami określonymi w [“Struktury” na stronie 1024](#).
- Pliki COPY o nazwach kończących się literą *H* są przeznaczone do użytku z programami, które używają łączenia statycznego, ale **nie** są inicjowane.
- Pliki COPY o nazwach kończących się literą *R* są przeznaczone do użytku z programami, które korzystają z dynamicznego łączenia. Te pliki są inicjowane z wyjątkami określonymi w [“Struktury” na stronie 1024](#).

Pliki COPY znajdują się w bibliotece QRPGLSRC w bibliotece QMQM.

Dla każdego zestawu plików COPY znajdują się dwa pliki zawierające nazwane stałe i jeden plik dla każdej z tych struktur. Pliki COPY są podsumowywane w programie [Tabela 123 na stronie 1022](#).

Tabela 123. Pliki RPG COPY			
Nazwa pliku (powiązanie statyczne, zainicjowane, CMQ* G)	Nazwa pliku (powiązanie statyczne, niezainicjowane, CMQ* H)	Nazwa pliku (połączenie dynamiczne, zainicjowane, CMQ* R)	Spis treści
CMQBOG	CMQBOH	-	Struktura opcji begin
CMQCDG	CMQCDH	CMQCDR	Struktura definicji kanału

Tabela 123. Pliki RPG COPY (kontynuacja)

Nazwa pliku (powiązanie statyczne, zainicjowane, CMQ* G)	Nazwa pliku (powiązanie statyczne, niezainicjowane, CMQ* H)	Nazwa pliku (połączenie dynamiczne, zainicjowane, CMQ* R)	Spis treści
CMQCFBFG	CMQCFBFH	-	Parametr filtru bitowego PCF
CMQCFG	-	-	Stałe dla PCF i zdarzeń
CMQCFBSG	CMQCFBSH	-	Łańcuch bajtowy PCF
CMQCFGRG	CMQCFGRH	-	PCF, parametr grupy
CMQCFIFG	CMQCFIFH	-	Parametr filtru liczby całkowitej PCF
CMQCFHG	CMQCFHH	-	Nagłówek PCF
CMQCFILG	CMQCFILH	-	Struktura parametru listy całkowitoliczbowej PCF
CMQCFING	CMQCFINH	-	Struktura parametru liczby całkowitej PCF
CMQCFSG	CMQCFSH	-	Parametr filtru łańcucha PCF
CMQCFSLG	CMQCFSLH	-	Struktura parametru listy łańcuchów PCF
CMQCFSTG	CMQCFSTH	-	Struktura parametru łańcucha PCF
CMQCFXLG	CMQCFXLH	-	Skrócona nazwa PCF dla CFIL64
CMQCFXNG	CMQCFXNH	-	Skrócona nazwa PCF dla CFIN64
CMQCIHG	CMQCIHH	-	Struktura nagłówka informacyjnego produktu CICS
CMQCNQG	CMQCNQH	-	Struktura opcji łączenia
CMQCSPG	CMQCSPH	-	Parametry bezpieczeństwa
CMQCXPG	CMQCXPH	CMQCXPR	Struktura parametru wyjścia kanału
CMQDHG	CMQDHH	CMQDHR	Struktura nagłówka dystrybucji
CMQDLHG	CMQDLHH	CMQDLHR	Struktura nagłówka martwej litery
CMQDXPG	CMQDXPH	CMQDXPR	Struktura parametru wyjścia konwersji danych
CMQEPHG	CMQEPHH	-	Wbudowana struktura nagłówka PCF
CMQG	-	CMQR	Stałe nazwane dla głównego interfejsu MQI
CMQGMQG	CMQGMQH	CMQGMOR	Pobierz strukturę opcji komunikatu
CMQIIHG	CMQIIHH	CMQIIHR	Struktura nagłówka informacyjnego produktu IMS
CMQMDEG	CMQMDEH	KMQMDER	Struktura rozszerzenia deskryptora komunikatu
CMQMDG	CMQMDH	CMQMDR	Struktura deskryptora komunikatu

Tabela 123. Pliki RPG COPY (kontynuacja)

Nazwa pliku (powiązanie statyczne, zainicjowane, CMQ* G)	Nazwa pliku (powiązanie statyczne, niezainicjowane, CMQ* H)	Nazwa pliku (połączenie dynamiczne, zainicjowane, CMQ* R)	Spis treści
CMQMD1G	CMQMD1H	CMQMD1R	Struktura deskryptora komunikatu wersja 1
CMQMD2G	CMQMD2H	-	Struktura deskryptora komunikatu wersja 2
CMQODG	CMQODH	CMQODR	Struktura deskryptora obiektu
CMQORG	CMQORH	CMQORR	Struktura rekordu obiektu
CMQPMOG	CMQPMOH	CMQPMOR	Struktura opcji umieszczania komunikatów
CMQPSG	-	-	Stałe dla publikowania/subskrybowania
CMQRFHG	CMQRFHH	-	Reguły i struktura nagłówka formatowania
CMQRFH2G	CMQRFH2H	-	Struktura reguł i formatowanie nagłówka 2
CMQRMHG	CMQRMHH	CMQRMHR	Struktura nagłówka komunikatu odwołania
CMQRRG	CMQRRH	CMQRRR	Struktura rekordu odpowiedzi
CMQTMCG	CMQTMCH	CMQTMCR	Struktura komunikatu wyzwalacza (format znakowy)
CMQTMCG2G	CMQTMCG2H	CMQTMCG2R	Struktura komunikatu wyzwalacza (format znakowy) wersja 2
CMQTMG	CMQTMH	CMQTMR	Struktura komunikatu wyzwalacza
CMQWIHG	CMQWIHH	-	Struktura nagłówka informacji o pracy
CMQXG	-	CMQXR	Stałe nazwane dla wyjścia konwersji danych
CMQXQHG	CMQXQHH	CMQXQHR	Struktura nagłówka kolejki transmisji

Wywołania

Wywołania są opisane za pomocą ich indywidualnych nazw.

Parametry wywołania

Niektóre parametry przekazywane do interfejsu MQI mogą mieć więcej niż jedną funkcję współbieżną. Dzieje się tak dlatego, że przekazywana wartość całkowita jest często testowana na ustawieniu poszczególnych bitów w polu, a nie na jej łącznej wartości. Pozwala to na 'dodanie' kilku funkcji razem i przekazanie ich jako jednego parametru.

Struktury

Wszystkie struktury produktu IBM MQ są zdefiniowane z wartościami początkowymi dla pól, z następującymi wyjątkami:

- Dowolna struktura z przyrostkiem H.
- MQTMC
- MQTMC2

Te wartości początkowe są zdefiniowane w odpowiedniej tabeli dla każdej struktury.

Deklaracje struktury nie zawierają instrukcji DS . Dzięki temu aplikacja może zadeklarować pojedynczą strukturę danych lub strukturę danych o wielu wystąpieniach, kodując instrukcję DS , a następnie za pomocą instrukcji /COPY w celu skopiowania w pozostałej części deklaracji:

```
D*..1.....2.....3.....4.....5.....6.....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD      DS              5
D/COPY CMQMDR
```

Stałe nazwane

Istnieje wiele wartości liczbowych i wartości znakowych, które umożliwiają wymianę danych między programem użytkowym a menedżerem kolejek. Aby ułatwić bardziej czytelne i spójne podejście do korzystania z tych wartości, zdefiniowane są dla nich stałe nazwane. Można użyć tych stałych nazwanych, a nie wartości, które reprezentują, ponieważ zwiększa to czytelność kodu źródłowego programu.

Gdy w programie dołączono plik COPY CMQG do definiowania stałych, kompilator języka RPG wyda wiele komunikatów o poziomie istotności-zero dla stałych, które nie są używane przez program; komunikaty te są łagodne i mogą być bezpiecznie ignorowane.

Procedury MQI

Podczas tworzenia programu przy użyciu wywołań związanych z ILE należy wykonać wiązanie z procedurami MQI. Procedury te są eksportowane z następujących programów usługowych w zależności od potrzeb:

QMQM/AMQZSTUB

Ten program usługowy udostępnia powiązania kompatybilności dla aplikacji napisanych przed programem MQSeries V5.1 , które nie wymagają dostępu do nowych możliwości udostępnionych w wersji 5.1. Sygnatura tego programu usługowego jest zgodna z sygnaturą zawartą w wersji 4.2.1.

QMQM/LIBMQM

Ten program usługowy zawiera powiązania jednowątkowe dla wersji 5.1 i nowszych. Podczas pisania aplikacji wielowątkowych, należy zapoznać się z poniższą sekcją.

QMQM/LIBMQM_R

Ten program usługowy zawiera powiązania wielowątkowe dla wersji 5.1 i nowszych. Podczas pisania aplikacji wielowątkowych, należy zapoznać się z poniższą sekcją.

QMQM/LIBMQIC

Ten program usługowy jest przeznaczony do wiązania aplikacji klienckich, które nie są wielowątkowe.

QMQM/LIBMQIC_R

Ten program usługowy służy do wiązania wielowątkowych aplikacji klienckich.

Aby utworzyć programy, należy użyć komendy CRTPGM . Na przykład następująca komenda tworzy pojedynczy program wielowątkowy, który korzysta z wywołań związanych z ILE:

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

Zagadnienia związane z wątkami

Kompilator języka RPG używany przez produkt IBM i jest częścią zestawu narzędzi programistycznych produktu WebSphere oraz produktu WebSphere Development Studio dla produktu IBM i i jest znany jako kompilator języka ILE RPG IV.

Ogólnie, programy RPG nie powinny korzystać z wielowątkowych programów serwisowych. Wyjątkami są programy RPG utworzone przy użyciu kompilatora ILE RPG IV i zawierające słowo kluczowe THREAD (*SERIALIZE) w specyfikacji sterującej. Jednak nawet jeśli te programy są wątkowo bezpieczne, należy zwrócić szczególną uwagę na ogólny projekt aplikacji, ponieważ produkt THREAD (*SERIALIZE) wymusza serializację procedur RPG na poziomie modułu, co może mieć negatywny wpływ na ogólną wydajność.

W przypadku, gdy programy RPG są używane jako wyjścia konwersji danych, muszą być one bezpieczne dla wątku i powinny być rekompilowane za pomocą kompilatora języka ILE RPG w wersji 4.4 lub nowszej, z THREAD (*SERIALIZE) określonym w specyfikacji sterującej.

Więcej informacji na temat wielowątkowości zawiera publikacja *IBM i IBM MQ Development Studio: ILE RPG Reference* oraz *IBM i IBM MQ Development Studio: ILE RPG Programmer's Guide*.

Kontrola transakcji

Funkcje punktu synchronizacji MQI MQCMIT i MQBACK są dostępne dla programów w języku ILE RPG działających w trybie normalnym. Te wywołania pozwalają programowi na zatwierdzanie i wycofanie zmian w zasobach MQ.

Wywołania MQCMIT i MQBACK nie są dostępne dla programów w języku ILE RPG, które działają w trybie zgodności. W przypadku tych programów należy użyć kodów operacji COMMIT i ROLBK.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Kodowanie połączeń powiązanych

Procedury MQI ILE są wymienione w sekcji [Tabela 124 na stronie 1026](#).

Tabela 124. Wywołania związane ILE RPG obsługiwane przez każdy program usługowy				
Nazwa połączenia	LIBMQM i LIBMQM_R	AMQZSTUB	AMQVSTUB	LIBMQIC i LIBMQIC_R
MQBACK	Y			Y
MQBEGIN	Y			Y
MQCMIT	Y			Y
MQCLOSE	Y	Y		Y
MQCONN	Y	Y		Y
MQCONNX	Y			Y
MQDISC	Y	Y		Y
MQGET	Y	Y		Y
MQINQ	Y	Y		Y
MQOPEN	Y	Y		Y
MQPUT	Y	Y		Y
MQPUT1	Y	Y		Y
MQSET	Y	Y		Y
MQXCNVC	Y		Y	Y

Aby skorzystać z tych procedur, należy wykonać następujące czynności:

1. Zdefiniuj procedury zewnętrzne w specyfikacji D' D. Są one dostępne w ramach elementu CMQG pliku COPY zawierającego stałe nazwane.
 2. Użyj kodu operacji CALLP, aby wywołać procedurę wraz z jej parametrami.
- Na przykład wywołanie MQOPEN wymaga uwzględnienia następującego kodu:

```

D*****
D** MQOPEN Call -- Open Object (From COPY file CMQG) **
D*****
D*
D*.1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN PR EXTPROC('MQOPEN')
D* Connection handle
D HCONN 10I 0 VALUE
D* Object descriptor
D OBJDSC 224A
D* Options that control the action of MQOPEN
D OPTS 10I 0 VALUE
D* Object handle
D HOBJ 10I 0
D* Completion code
D CMPCOD 10I 0
D* Reason code qualifying CMPCOD
D REASON 10I 0
D*

```

Aby wywołać tę procedurę, po zainicjowaniu różnych parametrów potrzebny jest następujący kod:

```

...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
C CALLP MQOPEN(HCONN : MQOD : OPTS : HOBJ :
C CMPCOD : REASON)

```

W tym przypadku struktura MQOD jest definiowana za pomocą elementu COPY CMQODG, który dzieli go na jego komponenty.

Konwencje z adnotacjami

Te ostatnie tematy w tej sekcji pokazują, w jaki sposób:

- Wywołania powinny być wywoływane
- Parametry powinny być deklarowane
- Należy zadeklarować różne typy danych

W wielu przypadkach parametry są tablicami lub łańcuchami znaków o wielkości, która nie jest stała. W przypadku tych wartości małe litery "n" są używane do reprezentowania stałej liczbowej. Jeśli deklaracja dla tego parametru jest zakodowana, wartość "n" musi być zastąpiona wartością liczbową wymaganą.

MQAIR-rekord informacji uwierzytelniającej

Struktura MQAIR reprezentuje rekord informacji uwierzytelniających.

Przegląd

Cel: Struktura MQAIR umożliwia aplikacji uruchomionej jako klient IBM MQ określenie informacji o uwierzytelniającym, który ma być używany dla połączenia klienckiego. Struktura jest parametrem wejściowym w wywołaniu MQCONN.

Zestaw znaków i kodowanie: Dane w programie MQAIR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT.

- [“Poła” na stronie 1028](#)
- [“Wartości początkowe” na stronie 1030](#)

- [“Deklaracja RPG” na stronie 1030](#)

Pola

Struktura MQAIR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

AICN (10-cyfrowa liczba całkowita ze znakiem)

Jest to albo nazwa hosta, albo adres sieciowy hosta, na którym działa serwer LDAP. Po tym może wystąpić opcjonalny numer portu, ujęty w nawiasy.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2387.

Domyślny numer portu to 389.

To jest pole wejściowe. Długość tego pola nadawana jest przez LNAICN. Wartością początkową tego pola jest puste znaki.

AITYP (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ informacji uwierzytelniających zawartych w rekordzie.

Wartość musi być następująca:

AITLDP

Unieważnienie certyfikatu przy użyciu serwera LDAP.

Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2386.

To jest pole wejściowe. Wartością początkową tego pola jest AITLDP.

AIPW (10-cyfrowa liczba całkowita ze znakiem)

Jest to hasło wymagane do uzyskania dostępu do serwera CRL LDAP.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Jeśli serwer LDAP nie wymaga hasła lub użytkownik pominie nazwę użytkownika LDAP, wartość *AIPW* musi mieć wartość NULL lub być pusta. Jeśli nazwa użytkownika LDAP zostanie pominięta, a *AIPW* nie ma wartości NULL ani nie ma wartości pustej, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2390.

To jest pole wejściowe. Długość tego pola jest podana przez LNLDPW. Początkowa wartość tego pola jest pusta.

AILUL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość w bajtach nazwy użytkownika LDAP, która jest adresowana w polu *AILUP* lub *AILUO*. Wartość musi być z zakresu od zera do LNDISN. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2389.

Jeśli używany serwer LDAP nie wymaga nazwy użytkownika, należy ustawić wartość tego pola na zero.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

AILUO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie (w bajtach) nazwy użytkownika LDAP od początku struktury MQAIR.

Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli wartość *LDAPUserNameLength* wynosi zero.

Można użyć opcji *LDAPUserNamePtr* lub *LDAPUserNameOffset*, aby określić nazwę użytkownika LDAP, ale nie obie te wartości. Szczegółowe informacje można znaleźć w opisie pola *LDAPUserNamePtr*.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

AILUP (10-cyfrowa liczba całkowita ze znakiem)

Jest to nazwa użytkownika LDAP.

Składa się ona z nazwy wyróżniającej użytkownika, który próbuje uzyskać dostęp do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość określona w polu *AILUL*, należy zakończyć ją znakiem o kodzie zero lub dopełniać odstępami do długości *AILUL*. Pole jest ignorowane, jeśli wartość *AILUL* wynosi zero.

Nazwę użytkownika LDAP można podać na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *AILUP*

W takim przypadku aplikacja może zadeklarować łańcuch, który jest oddzielony od struktury MQAIR, i ustawić parametr *AILUP* na adres tego łańcucha.

Należy rozważyć użycie produktu *AILUP* dla języków programowania, które obsługują typ danych wskaźnika w sposób przenośny dla różnych środowisk (na przykład język programowania w języku C).

- Za pomocą pola przesunięcia *AILUO*

W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą strukturę MQSCO, po której następuje tablica rekordów MQAIR, po których następuje łańcuch nazwy użytkownika LDAP, a następnie ustaw *AILUO* na przesunięcie odpowiedniej nazwy łańcucha nazwy od początku struktury MQAIR. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być zakwaterowana w tabeli MQLONG (najbardziej restrykcyjnym językiem programowania jest język COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

Należy rozważyć użycie produktu *AILUO* dla języków programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który może nie być przenośny dla różnych środowisk (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki, należy używać tylko jednej z następujących opcji: *AILUP* i *AILUO*; Wywołanie nie powiodło się z kodem przyczyny RC2388.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

AISID (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

AISIDV

Identyfikator rekordu informacji uwierzytelniającej.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest AISIDV.

AIVER (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

AIVER1

Rekord informacji uwierzytelniających Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

AIRVERC

Bieżąca wersja rekordu informacji uwierzytelniającej.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to AIVER1.

Wartości początkowe

Tabela 125. Początkowe wartości pól w MQAIR dla MQAIR		
Nazwa pola	Nazwa stałej	Wartość stałej
AISID	AISIDV	'AIR~'
AIVER	AIVERC	1
AITYP	AITLDP	1
AICN	Brak	Pusty łańcuch lub odstępy
AILUP	Brak	Pusty wskaźnik lub zerowe bajty
AILUO	Brak	0
AILUL	Brak	0
AIPW	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQAIR Structure
D*
D* Structure identifier
D AISID 1 4 INZ('AIR ')
D* Structure version number
D AIVER 5 8I 0 INZ(1)
D* Type of authentication information
D AITYP 9 12I 0 INZ(1)
D* Connection name of CRL LDAP server
D AICN 13 276 INZ
D* Address of LDAP user name
D AILUP 277 292* INZ(*NULL)
D* Offset of LDAP user name from start of MQAIR structure
D AILUO 293 296I 0 INZ(0)
D* Length of LDAP user name
D AILUL 297 300I 0 INZ(0)
D* Password to access LDAP server
D AIPW 301 332 INZ

```

MQBMHO-Opcje uchwytu buforu do obsługi komunikatów

Struktura definiująca bufor do opcji uchwytu komunikatu.

Przegląd

Przeznaczenie: Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki uchwyt komunikatów są generowane z buforów. Struktura jest parametrem wejściowym w wywołaniu MQBUFMH.

Zestaw znaków i kodowanie: Dane w tabeli MQBMHO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1031](#)
- [“Wartości początkowe” na stronie 1031](#)

- [“Deklaracja RPG” na stronie 1031](#)

Pola

Struktura MQBMHO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

BMSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura bufora do struktury uchwytu komunikatu-pole StrucId .

Jest to identyfikator struktury. Wartość musi być następująca:

BMSIDV

Identyfikator dla struktury uchwytu komunikatu dla buforu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest BMSIDV.

BMVER (10-cyfrowa liczba całkowita ze znakiem)

Bufor do struktury uchwytu komunikatu-pole Wersja.

Jest to numer wersji struktury. Wartość musi być następująca:

BMVER1

Numer wersji dla struktury uchwytu buforu do komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

BBMVERVC

Bieżąca wersja buforu do struktury uchwytu komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to BMVER1.

BMOPT (10-cyfrowa liczba całkowita ze znakiem)

Bufor do struktury uchwytu komunikatu-pole Opcje.

Możliwe wartości:

BMDLPR

Właściwości, które są dodawane do uchwytu komunikatu, są usuwane z buforu. Jeśli wywołanie nie powiedzie się, żadne właściwości nie zostaną usunięte.

Opcje domyślne: Jeśli nie jest potrzebna opisana opcja, należy użyć następującej opcji:

BMBRAK

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest BMDLPR.

Wartości początkowe

Tabela 126. Początkowe wartości pól w MQBMHO		
Nazwa pola	Nazwa stałej	Wartość stałej
BMSID	BMSIDV	' BMHO '
BMVER	BMVER1	1
BMOPT	BMBRAK	0

Deklaracja RPG

```

D* MQBMHO Structure
D*
D*
D* Structure identifier
D  BMSID              1      4      INZ(' BMHO ')
```

```

D*
D* Structure version number
D  BMVER          5          8I 0 INZ(1)
D*
D* Options that control the action of MQBUFMH
D  BMOPT          9          12I 0 INZ(1)

```

MQBO-opcje rozpoczęcia

Struktura MQBO umożliwia aplikacji określanie opcji związanych z tworzeniem jednostki pracy.

Przegląd

Cel: Struktura jest parametrem wejściowym/wyjściowym w wywołaniu komendy MQBEGIN.

Zestaw znaków i kodowanie: Dane w obiekcie MQBO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT.

- [“Pola” na stronie 1032](#)
- [“Wartości początkowe” na stronie 1033](#)
- [“Deklaracja RPG” na stronie 1033](#)

Pola

Struktura MQBO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

BOOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje, które sterują działaniem komendy MQBEGIN.

Wartość musi być następująca:

BOBRAK

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest BONONE.

BOSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

BOSIDV

Identyfikator struktury opcji begin-options.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest BOSIDV.

BOVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

BOVER1

Numer wersji dla struktury opcji begin-options.

Następująca stała określa numer wersji bieżącej wersji:

BOVERC

Bieżąca wersja struktury opcji begin-options.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to BOVER1.

Wartości początkowe

Tabela 127. Początkowe wartości pól w obiekcie MQBO		
Nazwa pola	Nazwa stałej	Wartość stałej
BOSID	BOSIDV	'B0¬¬'
BOVER	BOVER1	1
BOOPT	BOBRAK	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```
D*.1....:....2....:....3....:....4....:....5....:....6....:....7..  
D* MQBO Structure  
D*  
D* Structure identifier  
D BOSID 1 4 INZ('B0 ¬¬')  
D* Structure version number  
D BOVER 5 8I 0 INZ(1)  
D* Options that control the action of MQBEGIN  
D BOOPT 9 12I 0 INZ(0)
```

MQCBC-kontekst wywołania zwrotnego

Struktura opisująca procedurę zwrotną.

Przegląd

Przeznaczenie

Struktura MQCBC służy do określania informacji kontekstowych, które są przekazywane do funkcji zwrotnej.

Struktura jest parametrem wejściowym/wyjściowym w wywołaniu procedury konsumenta komunikatów.

Wersja

Bieżąca wersja komendy MQCBC to CBCV2.

Zestaw znaków i kodowanie

Dane w tabeli MQCBC znajdują się w zestawie znaków określonym przez atrybut menedżera kolejek produktu *CodedCharSetId* oraz kodowanie lokalnego menedżera kolejek nadawanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura ta znajduje się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1033](#)
- [“Wartości początkowe” na stronie 1039](#)
- [“Deklaracja RPG” na stronie 1039](#)

Pola

Struktura MQCBC zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

CBCBUFFLEN (10-cyfrowa liczba całkowita ze znakiem)

Bufor może być większy niż wartość zarówno MaxMsg(wartość długości) zdefiniowana dla konsumenta, jak i wartość ReturnedLength (w przypadku wartości MQGMO).

Struktura kontekstu wywołania zwrotnego-pole BufferLength .

Jest to długość (w bajtach) buforu komunikatów, który został przekazany do tej funkcji.

Rzeczywista długość komunikatu jest podana w polu DataLength .

Aplikacja może wykorzystać cały bufor do własnych celów przez czas trwania funkcji zwrotnej.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji obsługi wyjątku.

CBCCALLBA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CallbackArea .

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian z pola CBDCCALLBA w strukturze MQCBD, która jest parametrem w wywołaniu MQCB używanym do definiowania funkcji zwrotnej.

Zmiany w *CBCCALLBA* są zachowywane w wywołaniach funkcji zwrotnej dla *CBCHOBJ*. To pole nie jest współużytkowane z funkcjami wywołania zwrotnego dla innych uchwytów.

Jest to pole wejściowe/wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CBCCALLT (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CallType .

Pole zawierające informacje o tym, dlaczego ta funkcja została wywołana; zdefiniowane są następujące pola.

Typy wywołań dostarczania komunikatów: te typy wywołań zawierają informacje na temat komunikatu. Parametry *CBCLLEN* i *CBCBUFFLEN* są poprawne dla tych typów wywołań.

CBCTMR

Funkcja konsumenta komunikatów została wywołana z komunikatem, który został zniszczony w sposób destruktywny z uchwytu obiektu.

Jeśli wartością parametru *CBCCC* jest CCWARN, wartość pola *Reason* to RC2079 lub jeden z kodów wskazujących problem konwersji danych.

CBCTMN

Funkcja konsumenta komunikatów została wywołana z komunikatem, który nie został jeszcze zniszczony w wyniku destrukcyjnego usunięcia z uchwytu obiektu. Komunikat może zostać destruktywnie usunięty z uchwytu obiektu przy użyciu *MsgToken*.

Możliwe, że komunikat nie został usunięty, ponieważ:

- Opcje MQGMO zażądały operacji przeglądania, GMBR*
- Komunikat jest większy niż dostępny bufor, a opcje MQGMO nie określają gmatm

Jeśli wartością parametru *CBCCC* jest CCWARN, wartość pola *Reason* to RC2080 lub jeden z kodów wskazujących problem konwersji danych.

Typy wywołań kontroli zwrotnej: te typy połączeń zawierają informacje o kontroli wywołania zwrotnego i nie zawierają szczegółów dotyczących komunikatu. Te typy wywołań są wymagane przy użyciu komendy CBDOPT w strukturze MQCBD.

Parametry *CBCLLEN* i *CBCBUFFLEN* nie są poprawne dla tych typów wywołań.

CBCTRC

Celem tego typu wywołania jest umożliwienie wykonania pewnej początkowej konfiguracji przez funkcję zwrotną.

Funkcja zwrotna jest wywoływana natychmiast po zarejestrowaniu wywołania zwrotnego, to znaczy po powrocie z wywołania MQCB przy użyciu wartości pola *Operation* CBREG.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i procedur obsługi zdarzeń.

Jeśli jest to wymagane, jest to pierwsze wywołanie funkcji zwrotnej.

Wartością pola *CBCREA* jest RCNONE.

CBCTSC

Celem tego typu wywołania jest umożliwienie wykonania pewnej konfiguracji przez funkcję zwrotną po jej uruchomieniu, na przykład przywrócenie zasobów, które zostały wyczyszczone, gdy wcześniej została zatrzymana.

Funkcja zwrotna jest wywoływana, gdy połączenie jest uruchamiane przy użyciu CTLSR lub CTLSW.

Jeśli funkcja zwrotna jest zarejestrowana w innej funkcji wywołania zwrotnego, ten typ wywołania jest wywoływany po powrocie wywołania zwrotnego.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartością pola *CBCREA* jest RCNONE.

CBCTTC

Celem tego typu wywołania jest umożliwienie wykonania przez funkcję zwrotną niektórych procedur czyszczących po zatrzymaniu przez pewien czas, na przykład przy czyszczeniu dodatkowych zasobów, które zostały nabyte podczas korzystania z komunikatów.

Funkcja zwrotna jest wywoływana, gdy wywołanie MQCTL jest wysyłane za pomocą wartości pola *Operation* CTLSP.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartość pola *CBCREA* jest ustawiona w taki sposób, aby wskazywać przyczynę zatrzymania.

CBCTDC

Celem tego typu wywołania jest umożliwienie wykonania przez funkcję zwrotną ostatniego czyszczenia na końcu procesu konsumowania. Funkcja zwrotna jest wywoływana, gdy:

- Funkcja zwrotna jest wyrejestrowywana przy użyciu wywołania MQCB z BCUNR.
- Kolejka jest zamknięta, co powoduje wyrejestrowywanie niejawnie. W tym przypadku funkcja zwrotna jest przekazywana do HOUNUH jako uchwyt obiektu.
- Wywołanie MQDISC kończy działanie-powoduje niejawnie zamknięcie, a tym samym wyrejestrowywanie. W tym przypadku połączenie nie jest natychmiast rozłączone, a każda bieżąca transakcja nie została jeszcze zatwierdzona.

Jeśli którekolwiek z tych działań zostanie wykonane w samej funkcji zwrotnej, działanie zostanie wywołane po powrocie wywołania zwrotnego.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i procedur obsługi zdarzeń.

Jeśli jest to wymagane, jest to ostatnie wywołanie funkcji zwrotnej.

Wartość pola *CBCREA* jest ustawiona w taki sposób, aby wskazywać przyczynę zatrzymania.

CBCTEC

Funkcja procedury obsługi zdarzeń

Funkcja procedury obsługi zdarzeń została wywołana bez komunikatu, gdy:

- Wywołanie MQCTL jest wysyłane z wartością pola *Operation* CTLSP, lub
- Menedżer kolejek lub połączenia są zatrzymywane lub wyciszane.

To wywołanie może być użyte do podjęcia odpowiednich działań dla wszystkich funkcji zwrotnych.

- **Funkcja konsumenta komunikatów**

Funkcja konsumenta komunikatów została wywołana bez komunikatu, gdy wykryty został błąd (CBCCC = CCFAIL), który jest specyficzny dla uchwytu obiektu, na przykład CBCREA code = RC2016 .

Wartość pola CBCREA jest ustawiona w taki sposób, aby wskazywała przyczynę wywołania.

To jest pole wejściowe. CBCTMR i CMCTMN mają zastosowanie tylko do funkcji konsumenta komunikatów.

CBCCC (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CompCode .

Jest to kod zakończenia. Wskazuje, czy wystąpiły problemy z korzystaniem z komunikatu. Jest to jedna z następujących sytuacji:

CCOK

Pomyślne zakończenie

CCWARN

Ostrzeżenie (częściowe zakończenie)

CCFAIL

Wywołanie zakończone niepowodzeniem

To jest pole wejściowe. Wartością początkową tego pola jest CCOK.

CBCCONNAREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole ConnectionArea .

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian w polu ConnectionArea w strukturze MQCTLO. Jest to parametr wywołania MQCTL używanego do sterowania funkcją zwrotną.

Wszystkie zmiany wprowadzone w tym polu przez funkcje wywołania zwrotnego są zachowywane w obrębie wywołań funkcji zwrotnej. Ten obszar może być używany do przekazywania informacji, które mają być współużytkowane przez wszystkie funkcje wywołania zwrotnego. W przeciwieństwie do produktu *CallbackArea*, ten obszar jest wspólny dla wszystkich wywołań zwrotnych dla uchwytu połączenia.

Jest to pole wejściowe i wyjściowe. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CBCLLEN (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość w bajtach danych aplikacji w komunikacie. Jeśli wartość jest równa zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Pole CBCLLEN zawiera długość komunikatu, ale nie musi być długość danych komunikatu przekazywanych do konsumenta. Może to być komunikat, że komunikat został obcięty. Użyj pola [GMRL](#) w produkcie MQGMO, aby określić, ile danych zostało przekazane do konsumenta.

Jeśli kod przyczyny wskazuje, że komunikat został obcięty, można użyć pola CBCLLEN w celu określenia, jak duży jest rzeczywisty komunikat. Umożliwia to określenie wielkości buforu wymaganego do uwzględnienia danych komunikatu, a następnie wywołanie wywołania MQCB w celu zaktualizowania [CBDMMML](#) w tabeli MQCBD z odpowiednią wartością.

Jeśli zostanie podana opcja GMCONV, przekształcony komunikat może być większy niż wartość zwrócona przez wartość DataLength. W takich przypadkach aplikacja prawdopodobnie musi wydać wywołanie MQCB, aby zaktualizować [CBDMMML](#) w tabeli MQCBD, aby była większa niż wartość zwrócona przez menedżer kolejek dla DataLength.

Aby uniknąć problemów z obcinaniem komunikatów, należy podać wartość MaxMsg(MaxMsg) jako CBDFM. Powoduje to, że menedżer kolejek przydziela bufor dla pełnej długości komunikatu po konwersji danych. Należy jednak pamiętać, że nawet jeśli ta opcja jest określona, nadal możliwe

jest, że wystarczająca ilość pamięci masowej nie jest dostępna, aby poprawnie przetworzyć żądanie. Aplikacje powinny zawsze sprawdzać zwrócony kod przyczyny. Na przykład, jeśli nie jest możliwe przydzielenie wystarczającej ilości pamięci masowej w celu przekształcenia komunikatu, komunikaty są zwracane do nieprzekształconego aplikacji.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

CBCFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi zawierające informacje o tym konsumencie.

Zdefiniowana jest następująca opcja:

CBCFBE

Ta opcja może zostać zwrócona, jeśli poprzednie wywołanie MQCLOSE z użyciem opcji COQSC nie powiodło się z kodem przyczyny RC2458.

Kod ten wskazał, że jest zwracany ostatni komunikat z wyprzedzeniem i że bufor jest teraz pusty. Jeśli aplikacja wydaje inne wywołanie MQCLOSE za pomocą opcji COQSC, to zakończy się powodzeniem.

Należy zauważyć, że aplikacja nie ma gwarancji, że zostanie nadana komunikat z tym zestawem flag, ponieważ w buforze odczytu z wyprzedzeniem mogą być nadal komunikaty niezgodne z bieżącymi kryteriami wyboru. W tej instancji funkcja konsumenta jest wywoływana z kodem przyczyny RC2019.

Jeśli bufor odczytu z wyprzedzeniem jest pusty, konsument jest wywoływany z flagą CBCFBE, a kod przyczyny RC2518.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest to istotne dla funkcji procedury obsługi zdarzeń.

CBCHOBJ (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CBCHOBJ.

W przypadku wywołania do konsumenta komunikatów jest to uchwyt obiektu odnoszący się do konsumenta komunikatów.

W przypadku procedury obsługi zdarzeń ta wartość to HONONE.

Aplikacja może użyć tego uchwytu i znacznika komunikatu w bloku Opcje pobierania komunikatów, aby otrzymać komunikat, jeśli komunikat nie został usunięty z kolejki.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to HOUNUH

CBCRCD (10-cyfrowa liczba całkowita ze znakiem)

CBCRCD wskazuje, jak długo menedżer kolejek oczekuje przed podjęciem próby ponownego nawiązania połączenia. Pole może być modyfikowane przez procedurę obsługi zdarzeń w celu zmiany opóźnienia lub zatrzymania ponownego połączenia.

Pola **CBCRCD** należy używać tylko wtedy, gdy wartością pola **Reason** w kontekście Callback jest RC2545.

Przy wpisie do procedury obsługi zdarzeń wartością parametru **CBCRCD** jest liczba milisekund, przez które menedżer kolejek oczekuje przed podjęciem próby ponownego nawiązania połączenia. [Tabela 128 na stronie 1037](#) zawiera listę wartości, które można ustawić w celu zmodyfikowania zachowania menedżera kolejek po powrocie z procedury obsługi zdarzeń.

Tabela 128. CBCRCD wartości	
Wartość	Opis
-1	Nie podejmuje się dalszych prób ponownego połączenia. Do aplikacji zwracany jest błąd.
0	Spróbuj ponownie nawiązać połączenie natychmiast.

Tabela 128. CBCRCD wartości (kontynuacja)	
Wartość	Opis
>0	Poczekaj na tę liczbę milisekund przed ponowną próbą nawiązania połączenia.

CBCREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole Przyczyna.

Jest to kod przyczyny kwalifikujący *CBCCC*

To jest pole wejściowe. Wartością początkową tego pola jest RCNONE.

CBCSTATE (10-cyfrowa liczba całkowita ze znakiem)

Wskazanie stanu bieżącego konsumenta. To pole jest najbardziej wartościowane do aplikacji, gdy do funkcji konsumenta przekazywany jest niezerowy kod przyczyny.

Tego pola można użyć, aby uprościć programowanie aplikacji, ponieważ nie ma potrzeby tworzenia kodu dla każdego kodu przyczyny.

To jest pole wejściowe. Wartością początkową tego pola jest CSNONE.

Stan	Działanie menedżera kolejek	Wartość stałej
<i>CSNONE</i> Ten kod przyczyny reprezentuje normalne wywołanie bez dodatkowych informacji o przyczynie.	Brak; jest to normalne działanie.	0
<i>CSSUST</i> Te kody przyczyny reprezentują warunki tymczasowe.	Procedura zwrotna jest wywoływana w celu zgłoszenia warunku, a następnie zawieszenia. Po pewnym czasie system może ponownie podjąć próbę wykonania operacji, co może spowodować ponowne zwiększenie tego samego warunku.	1
<i>CSSUSU</i> Te kody przyczyny reprezentują warunki, w których wywołanie zwrotne musi działać w celu rozwiązania warunku.	Konsument jest zawieszony, a procedura zwrotna jest wywoływana w celu zgłoszenia warunku. Procedura zwrotna powinna rozstrzygać warunek, jeśli jest to możliwe, a także RESUME lub zamknąć połączenie.	2
<i>CSSUS</i> Te kody przyczyny reprezentują niepowodzenia, które uniemożliwiają dalsze wywołania zwrotne komunikatu.	Menedżer kolejek automatycznie zawiesza funkcję zwrotną. Jeśli funkcja zwrotna została wznowiona, prawdopodobnie ponownie otrzyma ten sam kod przyczyny.	3
<i>CSSTOP</i> Te kody przyczyny reprezentują koniec konsumpcji komunikatów.	Dostarczono do procedury obsługi wyjątków i do wywołań zwrotnych, które określono dla CBDTC. Dalsze komunikaty nie mogą być używane.	4

CBCSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole StrucId .

Jest to identyfikator struktury. Wartość musi być następująca:

CBCSI

Identyfikator struktury kontekstu wywołania zwrotnego.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest CBCSI.

CBCVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole Wersja.

Jest to numer wersji struktury. Wartość musi być następująca:

CBCV1

Struktura kontekstu wywołania zwrotnego Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

CBCCV

Bieżąca wersja struktury kontekstu wywołania zwrotnego.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to CBCV1.

Wartości początkowe

Tabela 129. Początkowe wartości pól w tabeli MQCBC		
Nazwa pola	Nazwa stałej	Wartość stałej
CBCSID	CBCSI	'CBC¬'
CBCVER	CBCV1	1
CBCCALLT	Brak	0
CBCHOBJ	HOUNUH	-1
CBCCALLBA	Brak	Pusty wskaźnik lub zerowe bajty
CBCCONNAREA	Brak	Pusty wskaźnik lub zerowe bajty
CBCCC	CCOK	0
CBCREA	RCBRAK	0
CBCSTATE	CSBRAK	0
CBCLEN	Brak	0
CBCBUFFLEN	Brak	0
CBCFLG	Brak	0
CBCRCD	brak	0

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```
D* MQCBC Structure
D*
D*
D* Structure identifier
D  CBCSID          1      4    INZ('CBC ')
D*
D* Structure version number
D  CBCVER          5      8I 0  INZ(1)
```

```

D*
D* Why Function was called
D  CBCCALLT          9      12I 0 INZ(0)
D*
D* Object Handle
D  CBCHOBJ           13      16I 0 INZ(-1)
D*
D* Callback data passed to the function
D  CBCCALLBA         17      32*  INZ(*NULL)
D*
D* MQCTL Data area passed to the function
D  CBCCONNAREA       33      48*  INZ(*NULL)
D*
D* Completion Code
D  CBCCC             49      52I 0 INZ(0)
D*
D* Reason Code
D  CBCREA            53      56I 0 INZ(0)
D*
D* Consumer State
D  CBCSTATE          57      60I 0 INZ(0)
D*
D* Message Data Length
D  CBCLEN            61      64I 0 INZ(0)
D*
D* Buffer Length
D  CBCBUFFLEN        65      68I 0 INZ(0)
D*
** Flags containing information about
D* this consumer
D  CBCFLG            69      72I 0 INZ(0)
D* Ver:1 **
D* Number of milliseconds before reconnect attempt
D  CBCRCD            73      76I 0 INZ(0)
D* Ver:2 **
D*

```

MQCBD-deskryptor wywołania zwrotnego

Struktura określająca funkcję zwrotną.

Przegląd

Cel: Struktura MQCBD służy do określania funkcji zwrotnej i opcji sterujących jej używaniem przez menedżer kolejek.

Struktura jest parametrem wejściowym w wywołaniu MQCB.

Wersja: Bieżąca wersja tabeli MQCBD to CBDV1.

Zestaw znaków i kodowanie: Dane w tabeli MQCBD muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one podawane przez atrybut menedżera kolejek produktu *CodedCharSetId* i ENNAT. Jeśli jednak aplikacja jest uruchomiona jako IBM MQ MQI client, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1040](#)
- [“Wartości początkowe” na stronie 1044](#)
- [“Deklaracja RPG” na stronie 1045](#)

Pola

Struktura MQCBD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CBDSCALLBA (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian z pola w strukturze MQCBD, która jest parametrem w deklaracji funkcji wywołania zwrotnego.

Wartość ta jest używana tylko na *Operation* o wartości CBREG, bez aktualnie zdefiniowanego wywołania zwrotnego, ale nie zastępuje poprzedniej definicji.

Jest to pole wejściowe i wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CBDSCALLBF (10-cyfrowa liczba całkowita ze znakiem)

Funkcja zwrotna jest wywoływana jako wywołanie funkcji.

Użyj tego pola, aby określić wskaźnik do funkcji zwrotnej.

Użytkownik *musi* podać wartość *CallbackFunction* lub *CallbackName*. W przypadku określenia obu wartości zwracany jest kod przyczyny RC2486 .

Jeśli nie zostanie ustawiona żadna wartość *CallbackName* ani *CallbackFunction* , wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2486.

Ta opcja nie jest obsługiwana w następujących środowiskach:

- CICSOn z/OS
- Języki programowania i kompilatory, które nie obsługują odwołań do funkcji-wskaźnik

W takich sytuacjach wywołanie kończy się niepowodzeniem z kodem przyczyny RC2486.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

CBDSCALLBN (10-cyfrowa liczba całkowita ze znakiem)

Funkcja zwrotna jest wywoływana jako dynamicznie połączony program.

Użytkownik *musi* podać wartość *CallbackFunction* lub *CallbackName*. W przypadku określenia obu wartości zwracany jest kod przyczyny RC2486 .

Jeśli wartość *CallbackName* lub *CallbackFunction* nie jest prawdziwa, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2486.

Moduł jest ładowany, gdy pierwsza procedura zwrotna do użycia jest zarejestrowana i rozładowana, gdy ostatnia procedura zwrotna ma używać jej deregisterów.

Poza tym, gdzie w poniższym tekście zaznaczono, że nazwa jest wyrównana do lewej strony w polu, bez odstępów wewnętrznych; sama nazwa jest dopełniona spacjami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

IBMi

Nazwa wywołania zwrotnego może mieć jeden z następujących formatów:

- Biblioteka "/" Program
- Library "/" ServiceProgram "("FunctionName")"

Na przykład: MyLibrary/MyProgram(MyFunction).

Nazwą biblioteki może być *LIBL. Nazwy bibliotek i programów są ograniczone do maksymalnie 10 znaków.

Systemy UNIX

Nazwa wywołania zwrotnego to nazwa dynamicznie ładowanego modułu lub biblioteki, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[path]library(function)
```

Jeśli ścieżka nie jest określona, używana jest ścieżka wyszukiwania systemu.

Długość nazwy jest ograniczona do 128 znaków.

Windows

Nazwa wywołania zwrotnego jest nazwą biblioteki dołączanej dynamicznie, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu i napędem:

```
[d:][path]library(function)
```

Jeśli napęd i ścieżka nie są określone, używana jest ścieżka wyszukiwania systemu.

Długość nazwy jest ograniczona do 128 znaków.

z/OS

Nazwa wywołania zwrotnego to nazwa modułu ładowalnego, który jest poprawny dla specyfikacji w parametrze EP makra LINK lub LOAD.

Długość nazwy jest ograniczona do 8 znaków.

z/OS CICS

Nazwa wywołania zwrotnego to nazwa modułu ładowalnego, który jest poprawny dla specyfikacji w parametrze PROGRAM makra komendy EXEC CICS LINK.

Długość nazwy jest ograniczona do 8 znaków.

Program może być zdefiniowany jako zdalny przy użyciu opcji REMOTESYTEM zainstalowanej definicji programu lub przez dynamiczny program routing.

Zdalny region CICS musi być połączony z programem IBM MQ, jeśli program ma używać wywołań funkcji API IBM MQ. Należy jednak zauważyć, że pole w strukturze MQCBC nie jest poprawne w systemie zdalnym.

Jeśli wystąpi błąd podczas próby załadowania programu *CallbackName*, do aplikacji zwracany jest jeden z następujących kodów błędów:

- RC2495
- RC2496
- RC2497

W dzienniku błędów zapisywany jest również komunikat zawierający nazwę modułu, dla którego podjęto próbę ładowania, oraz kod przyczyny niepowodzenia z systemu operacyjnego.

To jest pole wejściowe. Początkowa wartość tego pola jest łańcuchem pustym lub pustym.

CBDCALLBT (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ funkcji zwrotnej. Wartość musi być jedną z następujących wartości:

CBTMC

Definiuje tę procedurę zwrotną jako funkcję konsumenta komunikatów.

Funkcja zwrotna konsumenta komunikatu jest wywoływana, gdy komunikat, spełniający określone kryteria wyboru, jest dostępny na uchwycie obiektu i połączenie jest uruchamiane.

CBTEH

Definiuje tę procedurę zwrotną jako asynchroniczną procedurę zdarzeń. Nie jest ona kierowana do odbierania komunikatów dla uchwytu.

Program *Hobj* nie jest wymagany w wywołaniu obiektu MQCB definiującym procedurę obsługi zdarzeń i jest ignorowany, jeśli jest określony.

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko konsumenta komunikatów. Funkcja konsumenta jest wywoływana bez komunikatu, gdy wystąpi zdarzenie, na przykład menedżer kolejek lub połączenie zatrzymujące się lub wygaszające. Nie jest wywoływana dla warunków, które są specyficzne dla pojedynczego konsumenta komunikatów, na przykład RC2016.

Zdarzenia są dostarczane do aplikacji, niezależnie od tego, czy połączenie zostało uruchomione, czy zatrzymane, z wyjątkiem następujących środowisk:

- CICS w środowisku z/OS
- aplikacje wielowątkowe

Jeśli program wywołujący nie przekaże jednej z tych wartości, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2483 .

To jest zawsze pole wejściowe. Wartością początkową tego pola jest CBTMC.

CBDMML (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość (w bajtach) najdłuższego komunikatu, który może zostać odczytany z uchwytu i podany do procedury zwrotnej. Jeśli komunikat ma dłuższą długość, procedura zwrotna otrzymuje *MaxMsgLength* bajtów komunikatu, a kod przyczyny:

- RC2080 lub
- RC2079 , jeśli określono wartość GMATM.

Rzeczywista długość komunikatu jest podana w polu “CBCLEN (10-cyfrowa liczba całkowita ze znakiem)” na stronie 1036 struktury MQCBC.

Zdefiniowane są następujące wartości specjalne:

CBDFM

Długość buforu jest korygowana przez system w taki sposób, aby komunikaty były zwracane bez obcinania.

Jeśli dostępna jest niewystarczająca ilość pamięci, aby przydzielić bufor do odebrania komunikatu, system wywołuje funkcję zwrotną z kodem przyczyny RC2071 .

Jeśli na przykład zostanie wysłane żądanie konwersji danych, a ilość pamięci jest niewystarczająca do przekształcenia danych komunikatu, wówczas nieprzekształcony komunikat jest przekazywany do funkcji zwrotnej.

To jest pole wejściowe. Wartością początkową pola *MaxMsgLength* jest CBDFM.

CBDOPT (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole Opcje.

Można określić jedną lub wszystkie z poniższych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe). Należy zauważyć, że kombinacje, które nie są poprawne, są oznaczone; wszystkie pozostałe kombinacje są poprawne.

CBDFQ

Wywołanie MQCB nie powiedzie się, jeśli menedżer kolejek znajduje się w stanie wygaszania.

W systemie z/OS ta opcja wymusza również, aby wywołanie MQCB nie powiodło się, jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wygaszania.

Określ wartość GMFIQ w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadamianie konsumentów komunikatów, gdy są one wygaszane.

Opcje sterujące: Następujące opcje kontrolują, czy funkcja zwrotna jest wywoływana, bez komunikatu, kiedy stan zmienia się w konsumencie:

CBDRRC

Funkcja zwrotna jest wywoływana z wywołaniem typu CBCTRC

CBDSRC

Funkcja zwrotna jest wywoływana z wywołaniem typu CBCTSC.

CBDTTC

Funkcja zwrotna jest wywoływana z wywołaniem typu CBCTTC.

CBDDC

Funkcja zwrotna jest wywoływana z wywołaniem typu CBCTDC.

Więcej informacji na temat tych typów wywołań zawiera sekcja [“CBCCALLT \(10-cyfrowa liczba całkowita ze znakiem\)”](#) na stronie 1034 .

Opcja domyślna: Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

CBDNO

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

CBDNO jest zdefiniowane do dokumentacji programu pomocy; nie jest przeznaczone, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

To jest pole wejściowe. Wartością początkową pola *Options* jest CBDNO.

CBDSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole *StrucId* .

Jest to identyfikator struktury. Wartość musi być następująca:

CBDSI

Identyfikator struktury deskryptora wywołania zwrotnego.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest CBDSI.

CBDVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole *Wersja*.

Jest to numer wersji struktury. Wartość musi być następująca:

CBDV1

Struktura deskryptora wywołania zwrotnego *Version-1* .

Następująca stała określa numer wersji bieżącej wersji:

CBDCV

Bieżąca wersja struktury deskryptora wywołania zwrotnego.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to CBDV1.

Wartości początkowe

Tabela 130. Początkowe wartości pól w MQCBD		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	CBDSI	'CBD¬'
<i>Version</i>	CBDV1	1
<i>CallBackType</i>	CBTMC	1
<i>Options</i>	CBDNO	0
<i>CallBackArea</i>	Brak	Zerowe bajty
<i>CallBackFunction</i>	Brak	Zerowe bajty
<i>CallBackName</i>	Brak	Puste
<i>MaxMsgLength</i>	CBDPM	-1

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```
D* MQCBD Structure
D*
D*
D* Structure identifier
D  CBDSID              1      4      INZ('CBD ')
D*
D* Structure version number
D  CBDVER              5      8I 0 INZ(1)
D*
D* Callback function type
D  CBDCALLBT          9      12I 0 INZ(1)
D*
** Options controlling message
D* consumption
D  CBDOPT            13      16I 0 INZ(0)
D*
D* User data passed to the function
D  CBDCALLBA        17      32*
D*
D* FP: Callback function pointer
D  CBDCALLBF        33      48*
D*
D* Callback name
D  CBDCALLBN        49      176      INZ('\0')
D*
D* Maximum message length
D  CBDMML          177      180I 0 INZ(-1)
```

MQCHARV-łańcuch o zmiennej długości

Użyj struktury MQCHARV, aby opisać łańcuch o zmiennej długości.

Przegląd

Zestaw znaków i kodowanie: Dane w tabeli MQCHARV muszą znajdować się w kodowaniu lokalnego menedżera kolejek, który jest nadawany przez ENNAT, oraz zestaw znaków pola VCHRC w strukturze. Jeśli aplikacja jest uruchomiona jako IBM MQ MQI client, struktura musi znajdować się w kodowaniu klienta. Niektóre zestawy znaków mają reprezentację, która zależy od kodowania. Jeśli wartość VCHRC jest jednym z tych zestawów znaków, używane kodowanie jest takie samo, jak kodowanie innych pól w tabeli MQCHARV. Zestaw znaków identyfikowany przez VSCCSID może być zestawem znaków dwubajtowych (DBCS).

Użycie: Struktura MQCHARV odnosi się do danych, które mogą być nieciągłe ze strukturą zawierającą ją. W celu rozwiązania tych danych można użyć pól zadeklarowanych z typem danych wskaźnika.

- [“Pola” na stronie 1045](#)
- [“Wartości początkowe” na stronie 1047](#)
- [“Deklaracja RPG” na stronie 1047](#)
- [“Redefinicja CSAPL” na stronie 1047](#)

Pola

Struktura MQCHARV zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

VCHRC (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator zestawu znaków łańcucha o zmiennej długości, który jest adresowany przez pole VCHRP lub VCHRO.

Wartością początkową tego pola jest CSAPL. Wartość ta jest definiowana przez produkt IBM MQ w celu wskazania, że menedżer kolejek powinien zostać zmieniony na prawdziwy identyfikator zestawu znaków menedżera kolejek. Jest to w ten sam sposób jak CSQM zachowuje się. W wyniku tego wartość CSAPL nigdy nie jest powiązana z łańcuchem o zmiennej długości. Początkową wartość

tego pola można zmienić, definiując inną wartość dla stałej CSAPL dla danej jednostki kompilacji za pomocą odpowiednich środków dla języka programowania aplikacji.

VCHRL (10-cyfrowa liczba całkowita ze znakiem)

Długość (w bajtach) łańcucha zmiennej długości adresowana przez pole VCHRP lub VCHRO.

Wartością początkową tego pola jest 0. Wartość musi być większa lub równa zero lub następująca wartość specjalna jest rozpoznawana:

VSNLT

Jeśli parametr VSNLT nie jest określony, jako część łańcucha dołączane są bajty VCHRL. Jeśli występują znaki o wartości NULL, nie są one ograniczane przez łańcuch.

Jeśli określono wartość VSNLT, łańcuch jest ograniczany przez pierwsze pole puste napotkane w łańcuchu. Sama wartość NULL nie jest uwzględniana jako część tego łańcucha.

Uwaga: Znak o kodzie zero używany do zakończenia łańcucha, jeśli określony jest parametr VSNLT, to wartość NULL z zestawu kodowego określonego przez VCHRC.

Na przykład w kodowaniu UTF-16 (CCSIDUCS-2 1200 i 13488) jest to 2-bajtowe kodowanie Unicode, w którym wartość null jest reprezentowana przez 16-bitową liczbę wszystkich zer. W polu UTF-16 często spotykano pojedyncze bajty ustawione na wszystkie zero, które są częścią znaków (na przykład 7-bitowe znaki ASCII), ale łańcuchy będą zakończone znakiem o kodzie zero tylko wtedy, gdy na granicy parzystej bajtu zostaną znalezione dwa bajty "zero". Możliwe jest uzyskanie dwóch "zerowych" bajtów na granicy nieparzystej, gdy są one każdą częścią poprawnych znaków. Na przykład: x '01' x '00' x '00' x '30' reprezentuje dwa poprawne znaki Unicode i nie ma wartości null w przypadku zakończenia łańcucha.

VCHRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie w bajtach zmiennej długości zmiennej długości od początku tabeli MQCHARV lub struktury zawierającej ją.

Jeśli struktura MQCHARV jest osadzona w innej strukturze, ta wartość jest przesunięta w bajtach zmiennej długości łańcucha od początku struktury zawierającej tę strukturę MQCHARV. Jeśli struktura MQCHARV nie jest osadzona w innej strukturze, na przykład jeśli jest ona określona jako parametr w wywołaniu funkcji, to przesunięcie jest względne wobec początku struktury MQCHARV.

Przesunięcie może być dodatnie lub ujemne. Można użyć pola VCHRP lub VCHRO, aby określić łańcuch o zmiennej długości, ale nie oba.

Wartością początkową tego pola jest 0.

VCHRP (wskaźnik)

Jest to wskaźnik do łańcucha o zmiennej długości.

Można użyć pola VCHRP lub VCHRO, aby określić łańcuch o zmiennej długości, ale nie oba.

Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

VCHRS (10-cyfrowa liczba całkowita ze znakiem)

Wielkość buforu zajętego przez pole VCHRP lub VCHRO (w bajtach).

Jeśli struktura MQCHARV jest używana jako pole wyjściowe w wywołaniu funkcji, to pole musi zostać zainicjowane z długością udostępnionego buforu. Jeśli wartość VCHRL jest większa niż VCHRS, to tylko bajty VCHRS danych zostaną zwrócone do programu wywołującego w buforze.

Wartość musi być większa lub równa zero lub następująca wartość specjalna, która jest rozpoznawana:

VSUSL

Jeśli określona jest wartość VSUSL, długość buforu jest pobierana z pola VCHRL w strukturze MQCHARV. Ta wartość specjalna nie jest odpowiednia, jeśli struktura jest używana jako zmienna wyjściowa, a bufor jest udostępniany. Jest to początkowa wartość tego pola.

Wartości początkowe

Nazwa pola	Nazwa stałej	Wartość stałej
VCHRP	Brak	Pusty wskaźnik lub zerowa liczba bajtów.
VCHRO	Brak	0
VCHRS	VSUSL	-1
VCHRL	Brak	0
VCHRC	CSAPL	-3

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..  
D* MQCHARV Structure  
D*  
D* Address of variable length string  
D VCHRP          1      16*  
D* Offset of variable length string  
D VCHRO          17      20I 0  
D* Size of buffer  
D VCHRS          21      24I 0  
D* Length of variable length string  
D VCHRL          25      28I 0  
D* CCSID of variable length string  
D VCHRC          29      32I 0
```

Redefinicja CSAPL

W przeciwieństwie do języków programowania obsługiwanych na innych platformach, RPG nie ma sposobu na ponowne zdefiniowanie zdefiniowanej stałej, dlatego musisz ustawić każdy VCHRC specjalnie, jeśli chcesz użyć wartości innej niż CSAPL.

MQCIH-nagłówek CICS bridge

Struktura MQCIH opisuje informacje, które mogą być obecne na początku komunikatu wysyłanego do serwera CICS bridge za pomocą programu IBM MQ for z/OS.

Przegląd

Nazwa formatu: FMCICS.

Wersja: Bieżąca wersja produktu MQCIH to CIVER2. Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję produktu MQCIH, a wartość początkowa pola *CIVER* jest ustawiona na CIVER2.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQCIH i danych komunikatu aplikacji:

- Aplikacje, które łączą się z menedżerem kolejek, do którego należy kolejka CICS bridge, muszą udostępniać strukturę MQCIH, która znajduje się w zestawie znaków i kodowaniu menedżera kolejek. Wynika to z faktu, że konwersja danych struktury MQCIH nie jest wykonywana w tym przypadku.
- Aplikacje, które łączą się z innymi menedżerami kolejek, mogą udostępniać strukturę MQCIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań; konwersja MQCIH jest wykonywana przez odbierający agent kanału komunikatów połączony z menedżerem kolejek, do którego należy kolejka CICS bridge.

Uwaga: Jest do tego jeden wyjątek. Jeśli menedżer kolejek, który jest właścicielem kolejki produktu CICS bridge , używa produktu CICS do kolejkowania rozproszonego, wartość MQCIH musi znajdować się w zestawie znaków i kodowaniu menedżera kolejek, który jest właścicielem kolejki produktu CICS bridge .

- Dane komunikatu aplikacji zgodnie ze strukturą MQCIH muszą być w tym samym zestawie znaków i kodowaniu co struktura MQCIH. Pola *CICSI* i *CIENC* w strukturze MQCIH nie mogą być używane do określania zestawu znaków i kodowania danych komunikatu aplikacji.

Jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek, musi zostać dostarczone przez użytkownika wyjście konwersji danych w celu przekształcenia danych komunikatu aplikacji.

Użycie: Jeśli wartości wymagane przez aplikację są takie same, jak wartości początkowe przedstawione w programie [Tabela 132 na stronie 1057](#), a most jest uruchomiony z wartością AUTH=LOCAL lub AUTH=IDENTIFY, to struktura MQCIH może zostać pominięta w komunikacie. We wszystkich innych przypadkach struktura musi być obecna.

Most akceptuje strukturę MQCIH w wersji version-1 lub version-2 , ale w przypadku transakcji 3270 musi być używana struktura version-2 .

Aplikacja musi upewnić się, że pola udokumentowane jako pola "żądanie" mają odpowiednie wartości w komunikacie wysłanym do mostu. Pola te są danymi wejściowymi mostu.

Pola udokumentowane jako pola "odpowiedzi" są ustawiane przez CICS bridge w komunikacie odpowiedzi, który most wysyła do aplikacji. Informacje o błędach są zwracane w polach *CIRET*, *CIFNC*, *CICC*, *CIREAi* *CIAC* , ale nie wszystkie z nich są ustawiane we wszystkich przypadkach. [Tabela 131 na stronie 1048](#) pokazuje, które pola są ustawione dla różnych wartości *CIRET*.

<i>Tabela 131. Zawartość pól informacji o błędzie w strukturze MQCIH</i>				
<i>CIRET</i>	<i>CIFNC</i>	<i>CICC</i>	<i>CIREA</i>	<i>CIAC</i>
CRC000	-	-	-	-
CRC003	-	-	FBC*	-
CRC002 CRC008	Nazwa wywołania IBM MQ	IBM MQ <i>CMPCOD</i>	IBM MQ <i>REASON</i>	-
CRC001 CRC006 CRC007 CRC009	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	CICS ABCODE

- ["Pola" na stronie 1048](#)
- ["Wartości początkowe" na stronie 1057](#)
- ["Deklaracja RPG" na stronie 1058](#)

Pola

Struktura MQCIH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CIAC (4-bajtowy łańcuch znaków)

Kod abend.

Wartość zwracana w tym polu jest istotna tylko wtedy, gdy w polu *CIRET* znajduje się wartość CRC005 lub CRC004. Jeśli tak się stanie, *CIAC* zawiera wartość CICS ABCODE.

To jest pole odpowiedzi. Długość tego pola jest podana przez LNABNC. Początkowa wartość tego pola to 4 puste znaki.

Jest to indyktor określający, czy deskryptory ADS powinny być wysyłane w żądaniach SEND i RECEIVE BMS. Zdefiniowane są następujące wartości:

ADNONE

Nie wysyłaj ani nie odbieraj deskryptora ADS.

ADSEND

Wyślij deskryptor ADS.

ADRECV

Odbieranie deskryptora ADS.

ADMSGF

Użyj formatu komunikatu dla deskryptora ADS.

Powoduje to, że deskryptor ADS zostanie wysłany lub odebrany za pomocą długiej postaci deskryptora ADS. Długi formularz zawiera pola wyrównane w granicach 4-bajtowych.

Pole *CIADS* powinno być ustawione w następujący sposób:

- Jeśli deskryptory ADS *nie* są używane, należy ustawić pole na wartość *ADNONE*.
- Jeśli deskryptory ADS *są* używane, a w każdym środowisku z *tym samym* identyfikatorem CCSID, ustaw wartość pola na sumę *ADSEND* i *ADRECV*.
- Jeśli deskryptory ADS *są* używane, ale z *różnymi* identyfikatorami CCSID w każdym środowisku, ustaw pole na sumę wartości *ADSEND*, *ADRECV* i *ADMSGF*.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest *ADNONE*.

CIADS (10-cyfrowa liczba całkowita ze znakiem)

Wysyłanie/odbieranie deskryptora ADS.

Jest to indyktor określający, czy deskryptory ADS powinny być wysyłane w żądaniach *SEND* i *RECEIVE BMS*. Zdefiniowane są następujące wartości:

ADNONE

Nie wysyłaj ani nie odbieraj deskryptora ADS.

ADSEND

Wyślij deskryptor ADS.

ADRECV

Odbieranie deskryptora ADS.

ADMSGF

Użyj formatu komunikatu dla deskryptora ADS.

Powoduje to, że deskryptor ADS zostanie wysłany lub odebrany za pomocą długiej postaci deskryptora ADS. Długi formularz zawiera pola wyrównane w granicach 4-bajtowych.

Pole *CIADS* powinno być ustawione w następujący sposób:

- Jeśli deskryptory ADS *nie* są używane, należy ustawić pole na wartość *ADNONE*.
- Jeśli deskryptory ADS *są* używane, a w każdym środowisku z *tym samym* identyfikatorem CCSID, ustaw wartość pola na sumę *ADSEND* i *ADRECV*.
- Jeśli deskryptory ADS *są* używane, ale z *różnymi* identyfikatorami CCSID w każdym środowisku, ustaw pole na sumę wartości *ADSEND*, *ADRECV* i *ADMSGF*.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest *ADNONE*.

CIAI (4-bajtowy łańcuch znaków)

Klawisz AID.

Jest to początkowa wartość klucza AID podczas uruchamiania transakcji. Jest to wartość 1-bajtowa, wyrównana do lewej strony.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest nadawana przez *LNATID*. Wartością początkową tego pola jest 4 odstępy.

CIAUT (8-bajtowy łańcuch znaków)

Hasło lub passticket.

Jest to hasło lub passticket. Jeśli uwierzytelnianie za pomocą identyfikatora użytkownika jest aktywne dla partycji CICS bridge, produkt *CIAUT* jest używany z identyfikatorem użytkownika w kontekście tożsamości MQMD w celu uwierzytelnienia nadawcy komunikatu.

To jest pole żądania. Długość tego pola jest podana przez LNAUTH. Początkowa wartość tego pola wynosi 8 znaków odstępu.

CICC (10-cyfrowa liczba całkowita ze znakiem)

Kod zakończenia IBM MQ lub CICS EIBRESP.

Wartość zwracana w tym polu jest zależna od *CIRET* ; patrz [Tabela 131 na stronie 1048](#).

To jest pole odpowiedzi. Wartością początkową tego pola jest CCOK.

CICNC (łańcuch znakowy 4-bajtowy)

Kod transakcji abend.

Jest to kod abend używany do zakończenia transakcji (zwykle jest to transakcja konwersacyjna, która żąda większej ilości danych). W przeciwnym razie to pole jest puste.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest podana przez LNCNCL. Wartością początkową tego pola jest 4 odstępy.

CICP (10-cyfrowa liczba całkowita ze znakiem)

Pozycja kursora.

Jest to początkowa pozycja kursora, gdy transakcja jest uruchomiona. Później w przypadku transakcji konwersacyjnych pozycja kursora znajduje się w wektorze RECEIVE.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest obecne, jeśli wartość *CIVER* jest mniejsza niż *CIVER2*.

CICSI (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

CICT (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy zadanie może być konwersacyjne.

Jest to indyktor określający, czy zadanie powinno mieć uprawnienia do wydawania dodatkowych informacji, czy też powinno być wstrzymane. Wartość musi być jedną z następujących wartości:

CTYES

Zadanie jest konwersacyjne.

CTNO

Zadanie nie jest konwersacyjne.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest CTNO.

CIENC (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

CIEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie błędu w komunikacie.

Jest to pozycja niepoprawnych danych wykrytych przez wyjście mostu. To pole udostępnia przesunięcie od początku komunikatu do miejsca położenia niepoprawnych danych.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest obecne, jeśli wartość *CIVER* jest mniejsza niż *CIVER2*.

CIFAC (8-bajtowy łańcuch bitowy)

Znacznik obiektu mostu.

Jest to 8-bajtowy znacznik obiektu mostu. Celem znacznika obiektu pomostowego jest umożliwienie wielu transakcji w pseudokonwersacji korzystania z tego samego obiektu pomostowego (wirtualnego terminalu 3270). W przypadku pierwszego lub tylko komunikatu w pseudokonwersacji należy ustawić wartość *FCNONE*; wartość ta informuje program *CICS* o przydzielaniu nowego obiektu mostu dla tego komunikatu. Znacznik obiektu pomostowego jest zwracany w komunikatach odpowiedzi, gdy w komunikacie wejściowym podano niezerową wartość *CIFKT*. Kolejne komunikaty wejściowe mogą następnie używać tego samego znacznika obiektu pomostowego.

Zdefiniowane są następujące wartości specjalne:

FCBRAK

Nie określono tokenu *BVT*.

Jest to zarówno pole żądania, jak i pole odpowiedzi używane tylko dla transakcji 3270. Długość tego pola jest podana przez *LNFAAC*. Wartością początkową tego pola jest *FCNONE*.

CIFKT (10-cyfrowa liczba całkowita ze znakiem)

Czas zwolnienia obiektu mostu.

Jest to czas (w sekundach), przez który obiekt mostu będzie przechowywany po zakończeniu transakcji użytkownika. W przypadku transakcji niekonwersacyjnych wartość powinna wynosić zero.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0.

CIFL (4-bajtowy łańcuch znaków)

Atrybuty emulowane terminalu.

Jest to nazwa zainstalowanego terminalu, który ma być używany jako model dla obiektu pomostowego. Wartość pusta oznacza, że *CIFL* jest pobierana z definicji profilu transakcji mostu lub używana jest wartość domyślna.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest podana przez *LNFACL*. Wartością początkową tego pola jest 4 odstępy.

CIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Wartość musi być następująca:

CIFNON

Brak flag.

To jest pole żądania. Wartością początkową tego pola jest *CIFNON*.

CIFMT (8-bajtowy łańcuch znaków)

Nazwa formatu IBM MQ danych, które są następujące: *MQCIH*.

Określa nazwę formatu IBM MQ danych, które są zgodne ze strukturą *MQCIH*.

W wywołaniu *MQPUT* lub *MQPUT1* aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze *MQMD*.

Ta nazwa formatu jest również używana dla komunikatu odpowiedzi, jeśli pole *CIRFM* ma wartość *FMNONE*.

- W przypadku żądań *DPL CIFMT* musi być nazwą formatu komendy *COMMAREA*.
- W przypadku żądań 3270 *CIFMT* musi mieć wartość *CSQCBDCI*, a *CIRFM* musi mieć wartość *CSQCBDCO*.

Wyjścia konwersji danych dla tych formatów muszą być zainstalowane w menedżerze kolejek, w którym mają być uruchomione.

Jeśli komunikat żądania powoduje wygenerowanie komunikatu odpowiedzi o błędzie, komunikat odpowiedzi o błędzie ma nazwę formatu FMSTR.

To jest pole żądania. Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

CIFNC (4-bajtowy łańcuch znaków)

Nazwa wywołania IBM MQ lub funkcja CICS EIBFN.

Wartość zwracana w tym polu jest zależna od *CIRET* ; patrz Tabela 131 na stronie 1048. Następujące wartości są możliwe, gdy *CIFNC* zawiera nazwę wywołania IBM MQ :

CFCONN

Wywołanie MQCONN.

CFGET

Wywołanie MQGET.

CFINQ

Wywołanie MQINQ.

CFOPEN

Wywołanie MQOPEN.

CFPUT

Wywołanie MQPUT.

CFPUT1

Wywołanie MQPUT1 .

CFNONE

Brak połączenia.

To jest pole odpowiedzi. Długość tego pola jest podana przez LNFUNC. Wartością początkową tego pola jest CFNONE.

CIGWI (10-cyfrowa liczba całkowita ze znakiem)

Odstęp czasu oczekiwania dla wywołania MQGET wystawionego przez zadanie mostu.

To pole ma zastosowanie tylko wtedy, gdy parametr *CIUOW* ma wartość CUFRST. Umożliwia on aplikacji wysyłającej określenie przybliżonego czasu (w milisekundach), przez który wywołania MQGET wydane przez most powinny czekać na drugie i kolejne komunikaty żądań dla jednostki pracy uruchomionej przez ten komunikat. Przesłania to domyślny przedział czasu oczekiwania używany przez most. Mogą być stosowane następujące wartości specjalne:

WIDFLT

Domyślny interwał oczekiwania.

Spowoduje to, że program CICS bridge będzie oczekiwał na okres, który został określony podczas uruchamiania mostu.

WIULIM

Nieograniczony przedział czasu oczekiwania.

To jest pole żądania. Wartością początkową tego pola jest WIDFLT.

CIII (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 0. To pole nie jest obecne, jeśli wartość *CIVER* jest mniejsza niż CIVER2.

CILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQCIH.

Wartość musi być jedną z następujących wartości:

CILEN1

Długość struktury nagłówka informacyjnego produktu version-1 CICS .

CILEN2

Długość struktury nagłówka informacji o nazwie version-2 CICS .

Następująca stała określa długość bieżącej wersji:

CILENC

Długość bieżącej wersji struktury nagłówka informacyjnego produktu CICS .

To jest pole żądania. Początkowa wartość tego pola to CILEN2.

CILT (10-cyfrowa liczba całkowita ze znakiem)

Typ odsyłacza.

Wskazuje typ obiektu, który most powinien spróbować połączyć. Wartość musi być jedną z następujących wartości:

LTPROG

Program DPL.

LTTRAN

Transakcja 3270.

To jest pole żądania. Wartością początkową tego pola jest LTPROG.

CINTI (4-bajtowy łańcuch znaków)

Następna transakcja do załączenia.

Jest to nazwa następnej transakcji zwracanej przez transakcję użytkownika (zwykle przez EXEC CICS RETURN TRANSID). Jeśli nie ma następnej transakcji, to pole jest puste.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Długość tego pola jest podana przez LNTRID. Wartością początkową tego pola jest 4 odstęp.

CIODL (10-cyfrowa liczba całkowita ze znakiem)

Wyjściowa długość danych COMMAREA.

Jest to długość danych użytkownika, które mają zostać zwrócone do klienta w komunikacie odpowiedzi. Ta długość obejmuje 8-bajtową nazwę programu. Długość pola COMMAREA przekazana do programu dowiązanego jest wartością maksymalną tego pola i długością danych użytkownika w komunikacie żądania, pomniejszoną o 8.

Uwaga: Długość danych użytkownika w komunikacie jest długością komunikatu *wykluczając* strukturę MQCIH.

Jeśli długość danych użytkownika w komunikacie żądania jest mniejsza niż *CIODL*, używana jest opcja DATALENGTH komendy LINK . Dzięki temu LINK może działać wydajnie w innym regionie CICS .

Można użyć następującej wartości specjalnej:

OLINPT

Długość danych wyjściowych jest taka sama jak długość danych wejściowych.

Ta wartość może być potrzebna, nawet jeśli nie jest wymagana żadna odpowiedź, w celu zapewnienia, że komenda przekazana do programu połączonego jest wystarczająca.

Jest to pole żądania używane tylko w przypadku programów DPL. Początkowa wartość tego pola OLINPT.

CIREA (10-cyfrowa liczba całkowita ze znakiem)

IBM MQ przyczyna lub kod sprzężenia zwrotnego, lub CICS EIBRESP2.

Wartość zwracana w tym polu jest zależna od *CIRET* ; patrz [Tabela 131 na stronie 1048](#).

To jest pole odpowiedzi. Wartością początkową tego pola jest RCNONE.

CIRET (10-cyfrowa liczba całkowita ze znakiem)

Kod powrotu z mostu.

Jest to kod powrotu z CICS bridge opisujący wynik przetwarzania wykonywanego przez most. Pola *CIFNC*, *CICC*, *CIREAi* *CIAC* mogą zawierać dodatkowe informacje (patrz [Tabela 131 na stronie 1048](#)). Wartość ta jest jedną z następujących wartości:

CRC000

(0, X'000 ') Brak błędu.

CRC001

(1, X'001 ') Instrukcja EXEC CICS wykryła błąd.

CRC002

(2, X'002 ') Wywołanie IBM MQ wykryło błąd.

CRC003

(3, X'003 ') CICS bridge wykrył błąd.

CRC004

(4, X'004 ') CICS bridge zakończyła się nieprawidłowo.

CRC005

(5, X'005 ') Aplikacja zakończyła się nieprawidłowo.

CRC006

(6, X'006 ') Wystąpił błąd zabezpieczeń.

CRC007

(7, X'007 ') Program nie jest dostępny.

CRC008

(8, X'008 ') Komunikat drugi lub późniejszy w bieżącej jednostce pracy nie został odebrany w określonym czasie.

CRC009

(9, X'009 ') Transakcja nie jest dostępna.

To jest pole odpowiedzi. Początkowa wartość tego pola to CRC000.

CIRFM (8-bajtowy łańcuch znaków)

IBM MQ -nazwa formatu komunikatu odpowiedzi.

Jest to nazwa formatu produktu IBM MQ komunikatu odpowiedzi, który zostanie wysłany w odpowiedzi na bieżący komunikat. Reguły kodowania są takie same jak w przypadku pola *MDFMT* w strukturze MQMD.

Jest to pole żądania używane tylko w przypadku programów DPL. Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

CIRSI (4-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi wynosić 4 odstępy. Długość tego pola jest podana przez LNRSID.

CIRS1 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi zawierać 8 odstępów.

CIRS2 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi zawierać 8 odstępów.

CIRS3 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi zawierać 8 odstępów.

CIRS4 (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 0. To pole nie jest obecne, jeśli wartość *CIVER* jest mniejsza niż *CIVER2*.

CIRTI (4-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi wynosić 4 odstępów. Długość tego pola jest podana przez *LNTRID*.

CISC (4-bajtowy łańcuch znaków)

Kod początkowy transakcji.

Jest to indyktor określający, czy most emuluje transakcję terminalową, czy *START*. Wartość musi być jedną z następujących wartości:

SCSTRT

Uruchom.

DANE SCDANE

Uruchom dane.

SCTERM

Zakończ dane wejściowe.

BRAK

Brak.

W odpowiedzi z mostu pole to jest ustawione na kod początkowy odpowiedni dla następnego identyfikatora transakcji zawartego w polu *CINTI*. W odpowiedzi możliwe są następujące kody początkowe:

- SCSTRT
- DANE SCDANE
- SCTERM

W przypadku serwera transakcji *CICS* w wersji 1.2 to pole jest tylko polem żądania; jego wartość w odpowiedzi jest niezdefiniowana.

W przypadku produktu *CICS Transaction Server* w wersji 1.3 i kolejnych wersji jest to zarówno pole żądania, jak i pole odpowiedzi.

To pole jest używane tylko dla transakcji 3270. Długość tego pola jest podana przez *LNSTCO*. Wartością początkową tego pola jest *SCNONE*.

CISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

CISIDV

Identyfikator struktury nagłówka informacji *CICS*.

To jest pole żądania. Wartością początkową tego pola jest *CISIDV*.

CITES (10-cyfrowa liczba całkowita ze znakiem)

Status na końcu zadania.

W tym polu wyświetlany jest status transakcji użytkownika na końcu zadania. Zwracana jest jedna z następujących wartości:

TENOSY

Niezsynchronizowane.

Transakcja użytkownika nie została jeszcze zakończona i nie została zsynchronizowana. Pole *MDMT* w strukturze MQMD to MTRQST w tym przypadku.

TECMIT

Zatwierdź jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona, ale została ona zsynchronizowana z pierwszą jednostką pracy. Pole *MDMT* w strukturze MQMD to MTDGRM w tym przypadku.

TEBACK

Wycofuje jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona. Wycofana zostanie bieżąca jednostka pracy. Pole *MDMT* w strukturze MQMD to MTDGRM w tym przypadku.

TEENDT

Zadanie zakończenia.

Transakcja użytkownika została zakończona (lub została zakończona abkończyniem). Pole *MDMT* w strukturze MQMD to MTRPLY w tym przypadku.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Wartością początkową tego pola jest TENOSY.

CITI (4-bajtowy łańcuch znaków)

Transakcja do dołączenia.

Jeśli parametr *CILT* ma wartość LTTRAN, *CITI* jest identyfikatorem transakcji użytkownika, który ma zostać uruchomiony; w tym przypadku musi być określona wartość niepusta.

Jeśli parametr *CILT* ma wartość LTPROG, *CITI* jest kodem transakcji, w ramach którego mają być uruchamiane wszystkie programy w jednostce pracy. Jeśli podana wartość jest pusta, używany jest domyślny kod transakcji mostu DPL produktu CICS DPL (CKBP). Jeśli wartość jest niepusta, musi być zdefiniowana jako CICS jako lokalna transakcja TRANSACTION z programem początkowym CSQCBP00. To pole ma zastosowanie tylko wtedy, gdy wartość *CIUOW* ma wartość CUFRST lub CUONLY.

To jest pole żądania. Długość tego pola jest podana przez LNTRID. Wartością początkową tego pola jest 4 odstępy.

CIUOW (10-cyfrowa liczba całkowita ze znakiem)

Element sterujący jednostki pracy.

Steruje on przetwarzaniem jednostki pracy wykonywaną przez CICS bridge. Można zażądać utworzenia mostu w celu uruchomienia pojedynczej transakcji lub jednego lub większej liczby programów w ramach jednostki pracy. Pole wskazuje, czy produkt CICS bridge powinien uruchomić jednostkę pracy, wykonać żadaną funkcję w bieżącej jednostce pracy, czy zakończyć jednostkę pracy, zatwierdzając ją lub wycofując z niej. Obsługiwane są różne kombinacje, aby zoptymalizować przepływy transmisji danych.

Wartość musi być jedną z następujących wartości:

TYLKO KOSTKI

Uruchom jednostkę pracy, wykonaj funkcję, a następnie zatwierdź jednostkę pracy (DPL i 3270).

CUCONT

Dodatkowe dane dla bieżącej jednostki pracy (tylko 3270).

CUFRST

Uruchom jednostkę pracy i wykonaj funkcję (tylko DPL).

CUMIDL

Wykonywanie funkcji w ramach bieżącej jednostki pracy (tylko DPL).

CULAST

Wykonaj funkcję, a następnie zatwierdź jednostkę pracy (tylko DPL).

CUCMIT

Zatwierdź jednostkę pracy (tylko DPL).

CUBACK

Wycofuje się z jednostki pracy (tylko DPL).

To jest pole żądania. Wartością początkową tego pola jest CUONLY.

CIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

CIVER1

Struktura nagłówka informacji Version-1 CICS .

CIVER2

Struktura nagłówka informacji Version-2 CICS .

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

CIVERC

Bieżąca wersja struktury nagłówka informacyjnego produktu CICS .

To jest pole żądania. Początkowa wartość tego pola to CIVER2.

Wartości początkowe

Tabela 132. Początkowe wartości pól w MQCIH		
Nazwa pola	Nazwa stałej	Wartość stałej
CISID	CISIDV	'CIH~'
CIVER	CIVER2	2
CILEN	CILEN2	180
CIENC	Brak	0
CICSI	Brak	0
CIFMT	FMNONE	Puste
CIFLG	CIFNON	0
CIRET	CRC000	0
CICC	CCOK	0
CIREA	RCBRAK	0
CIUOW	TYLKO KOSTKI	273
CIGWI	WIDFLT	-2
CILT	LTPROG	1
CIODL	OLINPT	-1
CIFKT	Brak	0
CIADS	ADNONE	0

Tabela 132. Początkowe wartości pól w MQCIH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>CICT</i>	CTNO	0
<i>CITES</i>	TENOSY	0
<i>CIFAC</i>	FCBRAK	Wartości null
<i>CIFNC</i>	CFNONE	Puste
<i>CIAC</i>	Brak	Puste
<i>CIAUT</i>	Brak	Puste
<i>CIRS1</i>	Brak	Puste
<i>CIRFM</i>	FMNONE	Puste
<i>CIRSI</i>	Brak	Puste
<i>CIRTI</i>	Brak	Puste
<i>CITI</i>	Brak	Puste
<i>CIFL</i>	Brak	Puste
<i>CAIAI</i>	Brak	Puste
<i>CISC</i>	BRAK	Puste
<i>CICNC</i>	Brak	Puste
<i>CINTI</i>	Brak	Puste
<i>CIRS2</i>	Brak	Puste
<i>CIRS3</i>	Brak	Puste
<i>CICP</i>	Brak	0
<i>CIEO</i>	Brak	0
<i>CIII</i>	Brak	0
<i>CIRS4</i>	Brak	0

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQCIH Structure
D*
D* Structure identifier
D  CISID          1      4    INZ('CIH ')
D* Structure version number
D  CIVER          5      8I 0 INZ(2)
D* Length of MQCIH structure
D  CILEN          9     12I 0 INZ(180)
D* Reserved
D  CIENC         13     16I 0 INZ(0)
D* Reserved
D  CICSI         17     20I 0 INZ(0)
D* MQ format name of data that followsMQCIH
D  CIFMT         21     28    INZ('      ')
D* Flags
D  CIFLG         29     32I 0 INZ(0)

```

```

D* Return code from bridge
D CIRET 33 36I 0 INZ(0)
D* MQ completion code or CICSEIBRESP
D CICC 37 40I 0 INZ(0)
D* MQ reason or feedback code, or CICSEIBRESP2
D CIREA 41 44I 0 INZ(0)
D* Unit-of-work control
D CIUOW 45 48I 0 INZ(273)
D* Wait interval for MQGET call issued by bridge task
D CIGWI 49 52I 0 INZ(-2)
D* Link type
D CILT 53 56I 0 INZ(1)
D* Output COMMAREA data length
D CIODL 57 60I 0 INZ(-1)
D* Bridge facility release time
D CIFKT 61 64I 0 INZ(0)
D* Send/receive ADS descriptor
D CIAADS 65 68I 0 INZ(0)
D* Whether task can be conversational
D CICT 69 72I 0 INZ(0)
D* Status at end of task
D CITES 73 76I 0 INZ(0)
D* Bridge facility token
D CIFAC 77 84 INZ(X'00000000000000-00')
D
D* MQ call name or CICS EIBFN function
D CIFNC 85 88 INZ(' ')
D* Abend code
D CIAC 89 92 INZ
D* Password or passticket
D CIAUT 93 100 INZ
D* Reserved
D CIRS1 101 108 INZ
D* MQ format name of reply message
D CIRFM 109 116 INZ(' ')
D* Remote CICS system id to use
D CIRSI 117 120 INZ
D* CICS RTRANSID to use
D CIRTI 121 124 INZ
D* Transaction to attach
D CITI 125 128 INZ
D* Terminal emulated attributes
D CIFL 129 132 INZ
D* AID key
D CIAI 133 136 INZ
D* Transaction start code
D CISC 137 140 INZ(' ')
D* Abend transaction code
D CICNC 141 144 INZ
D* Next transaction to attach
D CINTI 145 148 INZ
D* Reserved
D CIRS2 149 156 INZ
D* Reserved
D CIRS3 157 164 INZ
D* Cursor position
D CICP 165 168I 0 INZ(0)
D* Offset of error in message
D CIEO 169 172I 0 INZ(0)
D* Reserved
D CIII 173 176I 0 INZ(0)
D* Reserved
D CIRS4 177 180I 0 INZ(0)
D*

```

MQCMHO-opcje tworzenia uchwytu komunikatu

Struktura produktu **MQCMHO** umożliwia aplikacjom określanie opcji, które sterują sposobem tworzenia uchwytów komunikatów.

Przegląd

Przeznaczenie

Struktura jest parametrem wejściowym w wywołaniu **MQCRTMH**.

Zestaw znaków i kodowanie

Dane w programie **MQCMHO** muszą znajdować się w zestawie znaków aplikacji i kodowania aplikacji (ENNAT).

- [“Pola” na stronie 1060](#)
- [“Wartości początkowe” na stronie 1061](#)
- [“Deklaracja RPG” na stronie 1061](#)

Pola

Struktura **MQCMHO** zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

CMOPT (10-cyfrowa liczba całkowita ze znakiem)

Można określić jedną z następujących opcji:

WARTOŚĆ CMVAL

Gdy program **MQSETMP** jest wywoływany w celu ustawienia właściwości w tym uchwycie komunikatu, sprawdzana jest poprawność nazwy właściwości w celu upewnienia się, że:

- nie zawiera niepoprawnych znaków.
 - nie rozpoczyna się "JMS" ani "usr.JMS", z wyjątkiem następujących:
 - JMSCorrelationID
 - JMSReplyTo
 - JMSType
 - JMSXGroupID
 - JMSXGroupSeq
- Te nazwy są zastrzeżone dla właściwości produktu JMS .
- nie jest jednym z następujących słów kluczowych, w żadnej mieszance górnej lub małej:
 - "AND"
 - "BETWEEN"
 - "ESCAPE"
 - "FALSE"
 - "IN"
 - "IS"
 - "LIKE"
 - "NIE"
 - "NULL"
 - "LUB"
 - "PRAWDA"
 - nie zaczyna się "Ciało". lub "Element główny." (z wyjątkiem pliku "Root.MQMD.").

Jeśli właściwość ma wartość MQ-defined ("mq. *") i nazwa jest rozpoznawana, pola deskryptora właściwości są ustawiane na poprawne wartości dla właściwości. Jeśli właściwość nie zostanie rozpoznana, pole *Support* deskryptora właściwości jest ustawione na wartość **PDSUPO** (więcej informacji na ten temat zawiera sekcja [PDSUP](#)).

CMDEFV

Oznacza to, że występuje domyślny poziom sprawdzania poprawności nazw właściwości.

Domyślny poziom sprawdzania poprawności jest równoważny z poziomem określonym przez produkt **CMVAL**.

W przyszłej wersji można zdefiniować opcję administracyjną, która spowoduje zmianę poziomu sprawdzania poprawności, które będzie wykonywane po zdefiniowaniu składnika **CMDEFV**.

Jest to wartość domyślna.

CNOVA

Nie ma sprawdzania poprawności nazwy właściwości. Patrz opis produktu **CMVAL**.

Opcja domyślna: Jeśli nie jest wymagana żadna z opcji opisanych wcześniej w tej sekcji, można użyć następującej opcji:

CMNONE

Wszystkie opcje przyjmują wartości domyślne. Użyj tej wartości, aby wskazać, że nie określono żadnych innych opcji. Program **CMNONE** pomaga w dokumentacji programu; nie jest planowane używanie tej opcji razem z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **CMDEFV**.

CMSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

CMSIDV

Identyfikator struktury opcji tworzenia uchwytu komunikatu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **CMSIDV**.

CMVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

CMVER1

Version-1 -tworzenie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

CMVERC

Bieżąca wersja struktury opcji tworzenia uchwytu komunikatu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **CMVER1**.

Wartości początkowe

Tabela 133. Początkowe wartości pól w MQCMHO		
Nazwa pola	Nazwa stałej	Wartość stałej
CMSID	CMSIDV	' CMHO '
CMVER	CMVER1	1
CMOPT	CMDEFV	0

Deklaracja RPG

```
D* MQCMHO Structure
D*
D*
D* Structure identifier
D  CMSID          1      4      INZ(' CMHO ')
```

```

D*
D* Structure version number
D  CMVER          5          8I 0 INZ(1)
D*
D* Options that control the action of MQCRTMH
D  CMOPT          9          12I 0 INZ(0)

```

MQCNO-opcje połączenia

Struktura MQCNO umożliwia aplikacji określanie opcji związanych z połączeniem z lokalnym menedżerem kolejek.

Przegląd

Cel: Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQCONN.

Wersja: Bieżąca wersja programu MQCNO to CNVER5. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję obiektu MQCNO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *CNVER* ustawioną na CNVER1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *CNVER* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie: Dane w tabeli MQCNO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek nadawanego przez ENNAT.

- [“Pola” na stronie 1062](#)
- [“Wartości początkowe” na stronie 1067](#)
- [“Deklaracja RPG” na stronie 1067](#)

Pola

Struktura MQCNO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CNCCO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach struktury definicji kanału MQCD od początku struktury MQCNO.

CNCCP (wskaźnik)

Jest to wskaźnik do struktury definicji kanału MQCD.

CNCONID (24-bajtowy łańcuch znaków)

Unikalny identyfikator połączenia. To pole umożliwia menedżerowi kolejek niezawodne identyfikowanie procesu aplikacji poprzez przypisanie mu unikalnego identyfikatora podczas pierwszego łączenia się z menedżerem kolejek.

Aplikacje używają identyfikatora połączenia do celów korelacji podczas wykonywania wywołań PUT i GET. Wszystkie połączenia są przypisywane do identyfikatora przez menedżera kolejek, bez względu na to, w jaki sposób połączenie zostało nawiązane.

Istnieje możliwość użycia identyfikatora połączenia w celu wymuszenia zakończenia długotrwałego działania jednostki pracy. W tym celu należy podać identyfikator połączenia za pomocą komendy PCF 'Stop Connection' (Zatrzymaj połączenie) lub komendy MQSC STOP CONN (STOP CONN). Więcej informacji na temat korzystania z tych komend można znaleźć w sekcji odsyłaczy do stron pokrewnych.

Początkowa wartość pola wynosi 24 puste bajty.

CNCT (128-bajtowy łańcuch bitowy)

Jest to znacznik, który menedżer kolejek wiąże z zasobami, które mają wpływ na aplikację w trakcie tego połączenia.

Znacznik połączenia menedżera kolejek.

Każda instancja aplikacji lub aplikacji musi użyć innej wartości dla znacznika, aby menedżer kolejek mógł poprawnie przekształcić do postaci szeregowej dostęp do odpowiednich zasobów. Aby uzyskać więcej szczegółów, zapoznaj się z opisami opcji CN* CT*. Znacznik przestaje być poprawny, gdy aplikacja kończy działanie, lub wywołuje wywołanie MQDISC.

Jeśli znacznik nie jest wymagany, należy użyć następującej wartości specjalnej:

CTNONE

Nie określono znacznika połączenia.

Wartość jest binarna zero dla długości pola.

To jest pole wejściowe. Długość tego pola jest podana przez LNCTAG. Wartością początkową tego pola jest CTNONE. To pole jest ignorowane, jeśli parametr *CNVER* jest mniejszy niż CNVER3.

Pola ConnTag należy używać podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS .

CNOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące działaniem programu MQCONN.

Opcje powiązania

Opcje powiązania sterują typem używanego powiązania IBM MQ ; należy określić tylko jedną z następujących opcji:

CNSBND

Powiązanie standardowe.

Opcja powiązania standardowego powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ zachowuje integralność menedżera kolejek; oznacza to, że chroni menedżer kolejek przed błędnymi programami.

Parametru CNSBND należy użyć w sytuacjach, w których aplikacja mogła nie zostać w pełni przetestowana lub może być niewiarygodna lub nie może być wiarygodna. Wartością domyślną jest CNSBND .

CNSBND jest zdefiniowana w dokumentacji programu pomocy. Nie należy używać tej opcji z żadną inną opcją kontrolującymi typ używanego powiązania, ale ponieważ jego wartość wynosi zero, nie można wykryć takiego użycia.

Ta opcja jest obsługiwana we wszystkich środowiskach.

CNFBND

Szybkie wiązanie ścieżki.

Opcja wiązania krótkiej ścieżki powoduje, że aplikacja i agent lokalnego menedżera kolejek mogą być częścią tej samej jednostki wykonywania. Krótka ścieżka jest w przeciwieństwie do powiązania standardowego, w którym aplikacja i agent menedżera kolejek lokalnych działają w oddzielnych jednostkach wykonywania.

Parametr CNFBND jest ignorowany, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane, ponieważ nie określono opcji.

CNFBND może być korzystne w sytuacjach, gdy wiele procesów zużywa więcej zasobów niż ogólny zasób używany przez aplikację. Aplikacja, która używa powiązania krótkiej ścieżki, jest nazywana *zaufaną aplikacją*.

Podczas podejmowania decyzji o wykorzystaniu powiązania krótkiej ścieżki należy wziąć pod uwagę następujące ważne punkty:

- **Korzystanie z opcji CNFBND nie zapobiega modyfikowaniu lub uszkodzeniu komunikatów oraz innych obszarów danych należących do menedżera kolejek. Tej opcji należy używać tylko w sytuacjach, w których w pełni oceniono te problemy.**

- Aplikacja nie może używać sygnałów asynchronicznych ani przerw licznika czasu (takich jak `sigkill`) z CNFBND. Istnieją również ograniczenia dotyczące korzystania z segmentów pamięci współużytkowanej.
- Aplikacja nie może w żadnym momencie mieć więcej niż jednego wątku połączonego z menedżerem kolejek.
- Aplikacja musi używać wywołania MQDISC do rozłączenia się z menedżerem kolejek.
- Aplikacja musi zakończyć działanie przed zakończeniem menedżera kolejek za pomocą komendy `endmqm`.

Następujące punkty mają zastosowanie w przypadku użycia CNFBND w wskazanych środowiskach:

- W systemie IBM izadanie musi być uruchamiane w profilu użytkownika QMQM, który należy do grupy QMQMADM. Ponadto, program nie może zakończyć się nieprawidłowo, w przeciwnym razie mogą wystąpić nieprzewidywalne rezultaty.

Więcej informacji na temat implikacji korzystania z zaufanych aplikacji zawiera sekcja [Nawiązywanie połączenia z menedżerem kolejek przy użyciu wywołań MQCONN i MQCONNX](#) i [Ograniczenia dla zaufanych aplikacji](#).

CNSHBD

Powiązania współużytkowane.

Opcja powiązań współużytkowanych powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ zachowuje integralność menedżera kolejek; oznacza to, że chroni menedżer kolejek przed błędnymi programami. Jednak niektóre zasoby są współużytkowane przez aplikację i agenta lokalnego menedżera kolejek. Parametr CNSHBD jest ignorowany, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

CNIBND

Izolowane Powiązania.

Opcja powiązań izolowanych powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ zachowuje integralność menedżera kolejek; oznacza to, że chroni menedżer kolejek przed błędnymi programami. Proces aplikacji i agent lokalnego menedżera kolejek są odizolowane od siebie, ponieważ nie współużytkują zasobów. Parametr CNIBND jest ignorowany, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

Opcje współużytkowania uchwytu

Następujące opcje sterują współużytkowaniem uchwytów między różnymi wątkami (jednostkami przetwarzania równoległego) w ramach tego samego procesu. Można określić tylko jedną z tych opcji.

CNHSN

Brak współużytkowania uchwytu między wątkami.

Opcja Brak współużytkowania uchwytu między wątkami wskazuje, że uchwytów połączeń i obiektów mogą być używane tylko przez wątek, który spowodował przydzielenie uchwytu; to jest wątek, który wydał wywołanie MQCONN, MQCONN lub MQOPEN. Uchwytów nie mogą być używane przez inne wątki należące do tego samego procesu.

CNHSB

Obsługa szeregowego uchwytu między wątkami, z blokowaniem wywołań.

Opcja współużytkowania uchwytu szeregowego między wątkami, z blokowaniem wywołania, opcja wskazuje, że uchwytów połączeń i obiektów przydzielone przez jeden wątek procesu mogą być używane przez inne wątki należące do tego samego procesu. Jednak tylko jeden wątek w danym momencie może użyć dowolnego uchwytu, to znaczy,

że dozwolone jest tylko użycie numeru seryjnego uchwytu. Jeśli wątek próbuje użyć uchwytu, który jest już używany przez inny wątek, bloki wywołań (czeka), aż do momentu, gdy uchwyt stanie się dostępny.

CNHSNB

Obsługa szeregowego uchwytu między wątkami, bez blokowania wywołań.

Obsługa szeregowego uchwytu między wątkami, *bez* blokowania wywołań, opcja jest taka sama, jak w przypadku " *with blocking* ", z wyjątkiem tego, że jeśli uchwyt jest używany przez inny wątek, wywołanie kończy się natychmiast CCFAIL i RC2219 , a nie blokując, dopóki uchwyt nie stanie się dostępny.

Wątek może mieć zero lub jedno niewspółużytkowane uchwyt, plus zero lub więcej współużytkowanych uchwytów:

- Każde wywołanie MQCONN lub MQCONNX , które określa CNHSN , zwraca nowy niewspółużytkowany uchwyt w pierwszym wywołaniu i ten sam niewspółużytkowany uchwyt w kolejnych wywołaniach (zakładając, że nie można wywołać żadnego wywołania MQDISC). Kod przyczyny to RC2002 w przypadku wywołań drugiego i późniejszych.
- Każde wywołanie MQCONNX , które określa CNHSB lub CNHSNB , zwraca nowy współużytkowany uchwyt dla każdego wywołania.

Uchwytów obiektów dziedziczą te same właściwości współużytkowania, co uchwyt połączenia określony w wywołaniu MQOPEN , która utworzyła uchwyt obiektu. Ponadto jednostki pracy dziedziczą te same właściwości współużytkowania, co uchwyt połączenia używany do uruchamiania jednostki pracy. Jeśli jednostka pracy jest uruchamiana w jednym wątku przy użyciu współużytkowanego uchwytu, to jednostka pracy może być aktualizowana w innym wątku przy użyciu tego samego uchwytu.

Jeśli opcja współużytkowania uchwytu nie zostanie określona, wartość domyślna jest określana przez środowisko:

- W środowisku produktu Microsoft Transaction Server (MTS) wartość domyślna jest taka sama jak CNHSB.
- W innych środowiskach wartość domyślna jest taka sama jak CNHSN.

Opcje ponownego połączenia

Opcje ponownego połączenia określają, czy połączenie jest nawiązywane ponownie. Tylko połączenia klienckie są ponownie nawiązywane.

CNRCDF

Opcja ponownego połączenia jest tłumaczona na wartość domyślną. Jeśli wartość domyślna nie jest ustawiona, wartość tej opcji jest tłumaczona na DISABLED. Wartość opcji jest przekazywana do serwera i może być odpytywana przez produkt **PCF** i **MQSC**.

CNRC

Aplikacja może zostać ponownie połączona z dowolnym menedżerem kolejek zgodnym z wartością parametru MQCONNX **QMNAME** . Opcji CNRC należy używać tylko wtedy, gdy nie ma powinowactwa między aplikacją kliencką a menedżerem kolejek, z którym początkowo nawiązało połączenie. Wartość opcji jest przekazywana do serwera i może być odpytywana przez produkt **PCF** i **MQSC**.

CNRCD

Nie można ponownie nawiązać połączenia z aplikacją. Wartość opcji *nie* jest przekazywana do serwera.

CNRCQM

Aplikacja może być ponownie połączona tylko z menedżerem kolejek, z którym jest pierwotnie połączona. Tej wartości należy użyć, jeśli klient może być ponownie połączony, ale istnieje powinowactwo między aplikacją kliencką a menedżerem kolejek, z którym pierwotnie nawiązało połączenie. Wartość tę należy wybrać, jeśli klient ma automatycznie nawiązywać ponowne połączenie z instancją rezerwową menedżera kolejek o wysokiej

dostępności. Wartość opcji jest przekazywana do serwera i może być odpytywana przez produkt **PCF** i **MQSC**.

Należy użyć opcji CNRC, CNRCDi CNRCQM tylko dla połączeń klienckich. Jeśli opcje są używane dla połączenia powiązania, MQCONNX kończy się niepowodzeniem z kodem zakończenia, MQCC_FAILED i kodem przyczyny, MQRC_OPTIONS_ERROR.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

CNNONE

Nie określono żadnych opcji.

Parametr CNNONE jest zdefiniowany w dokumentacji programu pomocy. Ta opcja nie jest używana z żadną inną opcją CN*, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

CNSCO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach struktury MQSCO od początku struktury MQCNO.

To pole jest ignorowane, jeśli parametr CNVER jest mniejszy niż CNVER4.

CNSCP (wskaźnik)

Jest to adres struktury MQSCO.

To pole jest ignorowane, jeśli parametr CNVER jest mniejszy niż CNVER4.

CNSECPO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie parametrów zabezpieczeń. Przesunięcie struktury MQCSP używane do określania identyfikatora użytkownika i hasła.

Wartość może być dodatnia lub ujemna. Wartością początkową tego pola jest 0.

To pole jest ignorowane, jeśli parametr CNVER jest mniejszy niż CNVER5.

CNSECPP (wskaźnik)

Wskaźnik parametrów zabezpieczeń. Adres struktury MQCSP używany do określania identyfikatora użytkownika i hasła.

Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

To pole jest ignorowane, jeśli parametr CNVER jest mniejszy niż CNVER5.

CNSID (4-bajtowy łańcuch znaków)

Identyfikator struktury dla struktury MQCNO.

Wartość musi być następująca:

CNSIDV

Identyfikator struktury opcji łączenia.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest CNSIDV.

CNVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury dla struktury MQCNO.

Wartość musi być następująca:

CNVER5

Struktura opcji connect-options Version-5 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

Następująca stała określa numer wersji bieżącej wersji:

CNVERC

Bieżąca wersja struktury opcji połączenia.

Wartości początkowe

Nazwa pola	Nazwa stałej	Wartość stałej
CNSID	CNSIDV	'CNO↵
CNVER	CNVER5	1
CNOPT	CNBRAK	0
CNCCO	Brak	0
CNCCP	Brak	Pusty wskaźnik lub zerowe bajty
CNCT	CTNONE	Wartości null
CNSCP	Brak	Pusty wskaźnik lub zerowe bajty
CNSCO	Brak	0
CNCONID	Brak	Wartości null
CNSECPO	Brak	0
CNSECPP	Brak	Pusty wskaźnik lub zerowe bajty

1. Symbol $\neg$ reprezentuje pojedynczy pusty znak.

```
D*.1.....2.....3.....4.....5.....6.....7...
D*
D* MQCNO Structure
D*
D* Structure identifier
D CNSID              1          4      INZ('CNO ')
D* Structure version number
D CNVER              5          8I 0 INZ(1)
D* Options that control the action ofMQCONNXX
D CNOPT              9         12I 0 INZ(0)
D* Offset of MQCD structure for clientconnection
D CNCCO             13        16I 0 INZ(0)
D* Address of MQCD structure for clientconnection
D CNCCP             17       32*    INZ(*NULL)
D* Queue-manager connection tag
D CNCT               33       160     INZ(X'00000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-
                                00000000000000000000-')
D* Address of MQSCO structure forclient connection
D CNSCP             161       176*   INZ(*NULL)
D* Offset of MQSCO structure for clientconnection
D CNSCO             177       180I 0 INZ(0)
```

D*	Unique Connection Identifier			
D	CNCONID	181	204	INZ(X'0000000000000000-0000000000000000000000-000000000000')
D				
D*	Offset of MQCSP structure			
D	CNSECPO	205	208I 0	INZ(0)
D*	Address of MQCSP structure			
D	CNSECPP	209	224*	INZ(*NULL)

MQCSP-parametry zabezpieczeń

Podsumowanie struktury MQCSP dla produktu IBM MQ for IBM i.

Przegląd

Cel: Struktura MQCSP umożliwia usłudze autoryzacji uwierzytelnianie identyfikatora użytkownika i hasła. Struktury parametrów zabezpieczeń połączenia MQCSP określa się w wywołaniu MQCONN.

Zestaw znaków i kodowanie: Dane w produkcie MQCSP muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT.

- [“Pola” na stronie 1068](#)
- [“Wartości początkowe” na stronie 1069](#)
- [“Deklaracja RPG” na stronie 1070](#)

Pola

Struktura MQCSP zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

CSAUTH (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ uwierzytelniania do wykonania.

Poprawne wartości:

CSAN

Nie należy używać pól identyfikatora użytkownika i hasła.

CSAUIAP

Uwierzytelniaj pola ID użytkownika i hasła.

To jest pole wejściowe. Wartością początkową tego pola jest CSAN.

CSCPPL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość hasła, które ma być używane w uwierzytelnianiu.

Maksymalna długość hasła nie jest zależna od platformy. Jeśli długość hasła jest większa niż wartość dozwolona, żądanie uwierzytelniania nie powiedzie się i zostanie zakończona niepowodzeniem z kodem RC2035.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSCPPO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach hasła, które ma być używane w uwierzytelnianiu.

Przesunięcie może być dodatnie lub ujemne.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSCPPP (wskaźnik)

Jest to adres hasła, który ma być używany w uwierzytelnianiu.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty.

CSCSPUIL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość identyfikatora użytkownika, który ma być używany w uwierzytelnianiu.

Maksymalna długość identyfikatora użytkownika nie jest zależna od platformy. Jeśli długość identyfikatora użytkownika jest większa niż wartość dozwolona, żądanie uwierzytelniania nie powiedzie się i zostanie zakończona niepowodzeniem z kodem RC2035.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSCSPUIO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach identyfikatora użytkownika, który ma być używany w uwierzytelnianiu.

Przesunięcie może być dodatnie lub ujemne.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

CSCSPUIP (wskaźnik)

Jest to adres ID użytkownika, który ma być używany w uwierzytelnianiu.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość CSVER jest mniejsza niż CSVER5.

CSRE1 (4-bajtowy łańcuch znaków)

Pole zarezerwowane, wymagane do wyrównania wskaźnika w systemie IBM i.

To jest pole wejściowe. Początkowa wartość tego pola jest równa null.

CSRS2 (8-bajtowy łańcuch znaków)

Pole zarezerwowane, wymagane do wyrównania wskaźnika w systemie IBM i.

To jest pole wejściowe. Początkowa wartość tego pola jest równa null.

CSSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

CSSIDV

Identyfikator struktury parametrów zabezpieczeń.

CSVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

CSVER1

Struktura parametrów zabezpieczeń Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

CVERC

Bieżąca wersja struktury parametrów zabezpieczeń.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to CSVER1.

Wartości początkowe

Tabela 135. Początkowe wartości pól w MQCNO		
Nazwa pola	Nazwa stałej	Wartość stałej
CSSID	CSSIDV	'CSP-'
CSVER	CSVER1	1

Tabela 135. Początkowe wartości pól w MQCNO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
CSAUTHT	Brak	0
CSRE1	Brak	Wartości null
CSCSPUIP	Brak	Wskaźnik pusty
CSCSPUIO	Brak	0
CSCSPUIL	Brak	0
CSRS2	Brak	Wartości null
CSCPPP	Brak	Wskaźnik pusty
CSCPP0	Brak	0
CSCPPL	Brak	0

Uwaga:

1. Symbol – reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQCSP Structure
D*
D* Structure identifier
D  CSSID              1      4      INZ('CSP ')
D* Structure version number
D  CSVER              5      8I 0 INZ(1)
D* Type of authentication
D  CSAUTHT            9      12I 0 INZ(0)
D* Reserved
D  CSRE1             13      16      INZ(X'00000000')
D* Address of user ID
D  CSCSPUIP          17      32*      INZ(*NULL)
D* Offset of user ID
D  CSCSPUIO          33      36I 0 INZ(0)
D* Length of user ID
D  CSCSPUIL          37      40I 0 INZ(0)
D* Reserved
D  CSRS2             41      48      INZ(X'0000000000000000')
D* Address of password
D  CSCPPP            49      64*      INZ(*NULL)
D* Offset of password
D  CSCPP0            65      68I 0 INZ(0)
D* Length of password
D  CSCPPL            69      72I 0 INZ(0)

```

MQCTLO-Struktura opcji wywołania zwrotnego elementu sterującego

Struktura określająca funkcję wywołania zwrotnego elementu sterującego.

Przegląd

Przeznaczenie

Struktura MQCTLO służy do określania opcji odnoszących się do funkcji zwrotnej sterowania.

Struktura jest parametrem wejściowym i wyjściowym w wywołaniu komendy [MQCTL](#) .

Wersja

Bieżącą wersją programu MQCTLO jest CTLV1.

Zestaw znaków i kodowanie

Dane w tabeli MQCTLO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1071](#)
- [“Wartości początkowe” na stronie 1072](#)
- [“Deklaracja RPG” na stronie 1072](#)

Pola

Struktura MQCTLO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

COCONNAREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji elementu sterującego-pole ConnectionArea.

Jest to pole, które jest dostępne dla funkcji zwrotnej, która ma być używana.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o zawartość tego pola i jest on przekazywany bez zmian z pola [CBCCONNAREA](#) w strukturze MQCBC, która jest parametrem w wywołaniu MQCB.

To pole jest ignorowane dla wszystkich operacji innych niż CTLSR i CTLSW.

Jest to pole wejściowe i wyjściowe do funkcji zwrotnej. Wartością początkową tego pola jest pusty wskaźnik lub zerowa liczba bajtów.

COOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje, które sterują działaniem MQCTLO.

CTLFQ

Wymuś niepowodzenie wywołania MQCTLO, jeśli menedżer kolejek lub połączenie znajduje się w stanie wygaszania.

Określ wartość GMFIQ w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadamianie konsumentów komunikatów, gdy są one wygaszane.

CTLTHR

Ta opcja informuje system, że aplikacja wymaga, aby wszystkie konsumenty komunikatów, dla tego samego połączenia, były wywoływane w tym samym wątku.

Opcja domyślna: Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

CTLNO

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. CTLNO jest zdefiniowane do dokumentacji programu pomocowego; nie jest przeznaczone, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

To jest pole wejściowe. Wartością początkową pola *COOPT* jest CTLNO.

CORSV (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem.

COSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji elementu sterującego-pole StrucId.

Jest to identyfikator struktury. Wartość musi być następująca:

CTLSI

Identyfikator struktury opcji sterowania.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest CTLSI.

COVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji sterowania-pole Wersja.

Jest to numer wersji struktury. Wartość musi być następująca:

CTLV1

Struktura opcji sterowania Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

CTLCV

Bieżąca wersja struktury opcji sterowania.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to CTLV1.

Wartości początkowe

Tabela 136. Początkowe wartości pól w MQCTLO		
Nazwa pola	Nazwa stałej	Wartość stałej
COSID	CTLSI	'CTLO '
COVER	CTLV1	1
COOPT	CTLNO	Wartości null
CORSV	Zarezerwowane pole	
COCONNAREA	Brak	Pusty wskaźnik lub zerowe bajty

Deklaracja RPG

```
D* MQCTLO Structure
D*
D*
D* Structure identifier
D COSID          1      4      INZ('CTLO')
D*
D* Structure version number
D COVER          5      8I 0 INZ(1)
D*
D* Options that control the action of MQCTL
D COOPT          9      12I 0 INZ(0)
D*
D* Reserved
D CORSV         13      16I 0 INZ(-1)
D*
D* MQCTL Data area passed to the function
D COCONNAREA    17      32*   INZ(*NULL)
```

MQDH-nagłówek dystrybucji

Struktura MQDH opisuje dodatkowe dane, które są obecne w komunikacie, gdy ten komunikat jest komunikatem listy dystrybucyjnej przechowywanej w kolejce transmisji.

Przegląd

Przeznaczenie: komunikat z listą dystrybucyjną jest komunikatem wysyłanym do wielu kolejek docelowych. Dodatkowe dane składają się z struktury MQDH, po której następuje tablica rekordów MQOR i tablica rekordów MQPMR.

Ta struktura jest używana przez wyspecjalizowane aplikacje, które umieszczają komunikaty bezpośrednio w kolejkach transmisji lub usuwają komunikaty z kolejek transmisji (na przykład: agenty kanałów komunikatów).

Ta struktura *nie* powinna być używana przez zwykłe aplikacje, które po prostu chcą umieszczać komunikaty w listach dystrybucyjnych. Aplikacje te powinny używać struktury MQOD do definiowania miejsc docelowych na liście dystrybucyjnej oraz struktury MQPMO w celu określenia właściwości komunikatu lub otrzymywania informacji o komunikatach wysyłanych do poszczególnych miejsc docelowych.

Zestaw znaków i kodowanie: Dane w MQDH muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek nadawanego przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie wartości MQDH muszą być ustawione w polach *MDCSI* i *MDENC* w:

- MQMD (jeśli struktura MQDH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQDH (wszystkie inne obserwacje).

Użycie: Gdy aplikacja umieszcza komunikat na liście dystrybucyjnej, a niektóre lub wszystkie miejsca docelowe są zdalne, menedżer kolejek prefikuje dane komunikatu aplikacji za pomocą struktur MQXQH i MQDH, a następnie umieszcza komunikat w odpowiedniej kolejce transmisji. W związku z tym dane są wykonywane w następującej kolejności, gdy komunikat znajduje się w kolejce transmisji:

- Struktura MQXQH
- Struktura MQDH plus tablice rekordów MQOR i MQPMR
- Dane komunikatu aplikacji

W zależności od miejsc docelowych menedżer kolejek może wygenerować więcej niż jeden taki komunikat i umieszczony w różnych kolejkach transmisji. W tym przypadku struktury MQDH w tych komunikatach identyfikują różne podzbiory miejsc docelowych zdefiniowanych przez listę dystrybucyjną otwieraną przez aplikację.

Aplikacja, która umieszcza komunikat z listą dystrybucyjną bezpośrednio w kolejce transmisji, musi być zgodna z opisaną wcześniej sekwencją i musi się upewnić, że struktura MQDH jest poprawna. Jeśli struktura MQDH jest niepoprawna, menedżer kolejek może nie powieść się z wywołania MQPUT lub MQPUT1 z kodem przyczyny RC2135.

Komunikaty mogą być zapisywane w kolejce w postaci listy dystrybucyjnej tylko wtedy, gdy kolejka jest zdefiniowana jako możliwa do obsługi komunikatów listy dystrybucyjnej (patrz atrybut kolejki *DistLists* opisany w sekcji „Atrybuty dla kolejek” na stronie 1395). Jeśli aplikacja umieszcza komunikat z listą dystrybucyjną bezpośrednio w kolejce, która nie obsługuje listy dystrybucyjnej, menedżer kolejek rozdziela komunikat listy dystrybucyjnej na poszczególne komunikaty i zamiast niego umieszcza je w kolejce.

- [“Pola” na stronie 1073](#)
- [“Wartości początkowe” na stronie 1076](#)
- [“Deklaracja RPG” na stronie 1077](#)

Pola

Struktura MQDH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

DHCNT (10-cyfrowa liczba całkowita ze znakiem)

Liczba rekordów MQOR, które są obecne.

Definiuje liczbę miejsc docelowych. Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *DHCNT* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

DHCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych, który jest zgodny z rekordami MQOR i MQPMR.

Określa identyfikator zestawu znaków dla danych, które są zgodne z tablicami rekordów MQOR i MQPMR. Nie ma on zastosowania do danych znakowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT

Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

DHENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych, które są zgodne z rekordami MQOR i MQPMR.

Określa kodowanie numeryczne danych, które są zgodne z tablicami rekordów MQOR i MQPMR; nie ma zastosowania do danych liczbowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

DHFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi ogólne.

Można określić następującą opcję:

DHFNEW

Wygeneruj nowe identyfikatory komunikatów.

Ta opcja wskazuje, że nowy identyfikator komunikatu ma być generowany dla każdego miejsca docelowego na liście dystrybucyjnej. Wartość tę można ustawić tylko wtedy, gdy nie ma żadnych rekordów komunikatów umieszczonych w zestawie komunikatów lub gdy rekordy są obecne, ale nie zawierają pola *PRMID*.

Użycie tej opcji powoduje odrabianie identyfikatorów komunikatów aż do ostatniego możliwego momentu, a mianowicie momentu, w którym komunikat z listą dystrybucyjną zostanie ostatecznie podzielony na poszczególne komunikaty. Minimalizuje to ilość informacji sterujących, które muszą przepływać wraz z komunikatem listy dystrybucyjnej.

Gdy aplikacja wstawi komunikat do listy dystrybucyjnej, menedżer kolejek ustawia DHFNEW w MQDH, które generuje, gdy spełniony jest oba z następujących warunków:

- Brak rekordów umieszczania komunikatów udostępnionych przez aplikację lub udostępnione rekordy nie zawierają pola *PRMID* (Nie zawiera rekordów zawierających komunikat).
- Pole *MDMID* w strukturze MQMD to MINONE lub pole *PMOPT* w produkcie MQPMO zawiera identyfikator PMNMID.

Jeśli nie są wymagane żadne opcje, można określić następujące opcje:

DHFNON

Brak flag.

Ta stała wskazuje, że nie określono żadnych flag. DHFNON jest zdefiniowane w dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

Wartością początkową tego pola jest DHFNON.

DHFMT (8-bajtowy łańcuch znaków)

Formatuj nazwę danych, które są zgodne z rekordami MQOR i MQPMR.

Określa nazwę formatu danych, które są zgodne z tablicami rekordów MQOD i MQPMR (w zależności od tego, która z tych zdarzeń wystąpi jako ostatnia).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

DHLEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQDH oraz następujące rekordy MQOR i MQPMR.

Jest to liczba bajtów od początku struktury MQDH do początku danych komunikatu zgodnie z tablicami rekordów MQOR i MQPMR. Dane są wykonywane w następującej kolejności:

- Struktura MQDH
- Tablica rekordów MQOR
- Tablica rekordów MQPMR
- Dane komunikatu

Tablice rekordów MQOR i MQPMR są adresowane przez przesunięcia zawarte w strukturze MQDH. Jeśli te przesunięcia powodują nieużywane bajty między jedną lub większą liczbą struktury MQDH, tablicami rekordów i danymi komunikatu, te nieużywane bajty muszą być uwzględnione w wartości *DHLEN*, ale treść tych bajtów nie jest zachowywana przez menedżer kolejek. Jest ona poprawna dla tablicy rekordów MQPMR w celu poprzedzania tablicy rekordów MQOR.

Wartością początkową tego pola jest 0.

DHORO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu MQOR od początku wywołania MQDH.

To pole umożliwia przesunięcie w bajtach pierwszego rekordu w tablicy rekordów obiektów MQOR, które zawierają nazwy kolejek docelowych. W tej tablicy znajdują się rekordy *DHCNT*. Rekordy te (plus wszystkie bajty pominięte między pierwszym rekordem obiektu a poprzednim polem) są uwzględniane w długości podanej w polu *DHLEN*.

Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *DHORO* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

DHPRF (10-cyfrowa liczba całkowita ze znakiem)

Flagi wskazujące, które pola MQPMR są obecne.

Można podać zero lub więcej następujących opcji:

ID_PFMID

Pole identyfikatora komunikatu jest obecne.

PFCID

Pole identyfikatora korelacji jest obecne.

Identyfikator PFGID

Pole identyfikatora grupy jest obecne.

PFFB

Pole informacji zwrotnej jest obecne.

PFACC

Pole tokenu rozliczania jest obecne.

Jeśli nie ma żadnych pól MQPMR, można określić następujące elementy:

PFNONE

Nie istnieją pola rekordu komunikatu umieszczonego w komunikacie.

PFNONE jest zdefiniowane w dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest PFNONE.

DHPRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu MQPMR od początku wywołania MQDH.

To pole zawiera przesunięcie w bajtach pierwszego rekordu w tablicy rekordów komunikatów MQPMR, które zawierają właściwości komunikatu. Jeśli ta tablica jest obecna, w tej tablicy znajdują się rekordy *DHCNT*. Rekordy te (plus wszystkie bajty pominięte między pierwszym rekordem umieszczania komunikatu a poprzednim polem) są uwzględniane w długości podanej w polu *DHLEN*.

Rekordy umieszczania komunikatów są opcjonalne; jeśli nie są dostępne żadne rekordy, wartość *DHPRO* wynosi zero, a *DHPRF* ma wartość PFNONE.

Wartością początkową tego pola jest 0.

DHSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DHSIDV

Identyfikator struktury nagłówka dystrybucji.

Wartością początkową tego pola jest DHSIDV.

DHVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DHVER1

Numer wersji struktury nagłówka dystrybucji.

Następująca stała określa numer wersji bieżącej wersji:

DHVERC

Bieżąca wersja struktury nagłówka dystrybucji.

Początkowa wartość tego pola to DHVER1.

Wartości początkowe

Tabela 137. Początkowe wartości pól w MQDH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>DHSID</i>	DHSIDV	'DH- - '
<i>DHVER</i>	DHVER1	1
<i>DHLEN</i>	Brak	0
<i>DHENC</i>	Brak	0
<i>DHCSI</i>	CSUNDF	0
<i>DHFMT</i>	FMNONE	Puste
<i>DHFLG</i>	DHFNON	0
<i>DHPRF</i>	PFNONE	0
<i>DHCNT</i>	Brak	0
<i>DHORO</i>	Brak	0

Tabela 137. Początkowe wartości pól w MQDH (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
DHPRO	Brak	0

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQDH Structure
D*
D* Structure identifier
D  DHSID          1      4    INZ('DH ')
D* Structure version number
D  DHVER          5      8I 0 INZ(1)
D* Length of MQDH structure plusfollowing MQOR and MQPMR records
D  DHLEN          9     12I 0 INZ(0)
D* Numeric encoding of data that followsthe MQOR and MQPMR records
D  DHENC         13     16I 0 INZ(0)
D* Character set identifier of data thatfollows the MQOR and MQPMR
D* records
D  DHCSI         17     20I 0 INZ(0)
D* Format name of data that follows theMQOR and MQPMR records
D  DHFMT         21     28    INZ(' ')
D* General flags
D  DHFLG         29     32I 0 INZ(0)
D* Flags indicating which MQPMR fieldsare present
D  DHPRF         33     36I 0 INZ(0)
D* Number of MQOR records present
D  DHCNT         37     40I 0 INZ(0)
D* Offset of first MQOR record from startof MQDH
D  DHORO         41     44I 0 INZ(0)
D* Offset of first MQPMR record fromstart of MQDH
D  DHPRO         45     48I 0 INZ(0)

```

MQDLH -nagłówek Dead-letter

Przegląd

Przeznaczenie

Struktura MQDLH opisuje informacje, które prefikują dane komunikatu aplikacji do komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). W kolejce niedostarczonych komunikatów może dojść do komunikatu, ponieważ menedżer kolejek lub agent kanału komunikatów przekierował go do kolejki. Aplikacja może umieścić komunikat bezpośrednio w kolejce.

Nazwa formatu

FMDLH

Zestaw znaków i kodowanie

MQDLH może znajdować się na początku danych komunikatu aplikacji. Jeśli tak, pola w strukturze MQDLH znajdują się w zestawie znaków i kodowaniu podanym w polach MDCSI i MDENC . Jeśli nie, zestaw znaków i kodowanie są ustawiane przez pola MDCSI i MDENC w strukturze nagłówek, które poprzedzają MQDLH.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Użycie

Aplikacje, które umieszczają komunikaty bezpośrednio w kolejce niedostarczonych komunikatów, muszą prefikować dane komunikatu do struktury MQDLH i inicjować pola odpowiednimi wartościami.

Menedżer kolejek nie wymaga jednak, aby struktura MQDLH była obecna lub czy podano poprawne wartości dla pól.

Jeśli komunikat jest zbyt długi, aby można go było umieścić w kolejce niedostarczonych komunikatów, aplikacja musi rozważyć wykonanie jednej z następujących czynności:

- Obetnij dane komunikatu, aby zmieściły się w kolejce niedostarczonych komunikatów.
- Zapisz komunikat w pamięci dyskowej i umieść w kolejce niedostarczonych komunikatów komunikat o wyjątku, który wskazuje, że komunikat jest zbyt długi.
- Usuń komunikat i zwróć błąd do jego inicjatora. Jeśli komunikat jest komunikatem krytycznym. Odrzuć komunikat tylko wtedy, gdy wiadomo, że twórca nadal ma kopię komunikatu. Na przykład komunikat otrzymany przez agenta kanału komunikatów z kanału komunikacyjnego.

To, które z wyborów jest odpowiednie, zależy od projektu aplikacji.

Menedżer kolejek wykonuje specjalne przetwarzanie, gdy komunikat, który jest segmentem, jest umieszczany w strukturze MQDLH z przodu. Więcej szczegółów można znaleźć w opisie struktury produktu MQMDE .

- [“Umieszczanie komunikatów w kolejce niedostarczonych komunikatów” na stronie 1078](#)
- [“Pobieranie komunikatów z kolejki niedostarczonych komunikatów” na stronie 1079](#)
- [“Pola” na stronie 1079](#)
- [“Wartości początkowe” na stronie 1083](#)
- [“Deklaracja RPG” na stronie 1083](#)

Umieszczanie komunikatów w kolejce niedostarczonych komunikatów

Jeśli komunikat jest umieszczany w kolejce niedostarczonych komunikatów, struktura MQMD używana dla wywołania MQPUT lub MQPUT1 musi być taka sama, jak MQMD powiązana z komunikatem. MQMD jest zwykle zwracany przez wywołanie MQGET , z wyjątkiem następujących przypadków:

- Pola MDCSI i MDENC muszą być ustawione na dowolny zestaw znaków i kodowanie, które są używane dla pól w strukturze MQDLH .
- Pole MDFMT musi być ustawione na wartość FMDLH , aby wskazać, że dane rozpoczynają się od struktury MQDLH .
- Pola kontekstu, MDACC, MDAID, MDAOD, MDPAN, MDPAT, MDPD, MDPTi MDUID , muszą być ustawione przy użyciu opcji kontekstu odpowiedniej dla okoliczności:
 - Aplikacja umieszczająca w kolejce niedostarczonych komunikatów komunikat, który nie jest powiązany z żadnym poprzedzającym komunikatem, musi używać opcji PMDEFC . Opcja PMDEFC powoduje, że menedżer kolejek ustawi wszystkie pola kontekstu w deskrypcorze komunikatu na wartości domyślne.
 - Aplikacja serwera umieszczając w kolejce niedostarczonych komunikatów komunikat, który otrzymał, musi użyć opcji PMPASA , aby zachować oryginalne informacje o kontekście.
 - Aplikacja serwera umieszczając w kolejce niedostarczonych komunikatów odpowiedź na odebraną wiadomość musi korzystać z opcji PMPASI . Opcja PMPASI zachowuje informacje o tożsamości, ale ustawia informacje o pochodzeniu, które mają być informacjami o serwerze aplikacji serwera.
 - Agent kanału komunikatów umieszczający w kolejce niedostarczonych komunikatów komunikat, który został odebrany z kanału komunikacyjnego, musi używać opcji PMSETA . Opcja PMSETA zachowuje oryginalną informację kontekstową.

W samej strukturze MQDLH pola muszą być ustawione w następujący sposób:

- Pola DLCSI, DLENCi DLFMT muszą być ustawione na wartości opisujące dane, które są zgodne ze strukturą MQDLH . Te wartości są zwykle wartościami z oryginalnego deskryptora komunikatu.
- Pola kontekstu DLPAT, DLPAN, DLPDi DLPT muszą być ustawione na wartości odpowiednie dla aplikacji, która umieszcza komunikat w kolejce niedostarczonych komunikatów. Wartości te nie są powiązane z pierwotnym komunikatem.

- Inne pola muszą być odpowiednio ustawione.

Aplikacja musi upewnić się, że wszystkie pola mają poprawne wartości, a pola znakowe są dopełniane spacjami do zdefiniowanej długości pola. Dane znakowe nie mogą być przerywane przedwcześnie, używając znaku o kodzie zero. Menedżer kolejek nie przekształca wartości NULL i kolejnych znaków w spacje w strukturze MQDLH.

Pobieranie komunikatów z kolejki niedostarczonych komunikatów

Aplikacje, które dostają komunikaty z kolejki niedostarczonych komunikatów, muszą sprawdzić, czy komunikaty zaczynają się od struktury MQDLH. Aplikacja może określić, czy struktura MQDLH jest obecna, badając pole MDFMT w deskrypcorze komunikatu MQMD. Jeśli w polu znajduje się wartość FMDLH, dane komunikatu zaczynają się od struktury MQDLH. Komunikaty w kolejce niedostarczonych komunikatów mogą zostać obcięte, jeśli początkowo były zbyt długie dla kolejki, dla której były przeznaczone.

Pola

Struktura MQDLH zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

DLCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków dla danych, które są następujące MQDLH.

DLCSI -określa identyfikator zestawu znaków dla danych, które są zgodne ze strukturą MQDLH. Dane zwykle pochodzą z oryginalnego komunikatu. Nie ma on zastosowania do danych znakowych w samej strukturze MQDLH.

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CSINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych po tej strukturze znajdują się w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli nie wystąpi błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Produkt CSINHT nie może być używany, jeśli wartością pola MDPAT w produkcie MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

DLDM (48-bajtowy łańcuch znaków)

Nazwa oryginalnego docelowego menedżera kolejek.

Jest to nazwa menedżera kolejek, który był oryginalnym miejscem docelowym dla komunikatu.

Długość tego pola jest podawana przez produkt LNQM. Początkowa wartość tego pola to 48 znaków odstępu.

DLDQ (48-bajtowy łańcuch znaków)

Nazwa oryginalnej kolejki docelowej.

Jest to nazwa kolejki komunikatów, która była oryginalnym miejscem docelowym dla komunikatu.

Długość tego pola jest podawana przez produkt LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

DLENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych, które są następujące MQDLH.

DLENC określa kodowanie liczbowe dla danych, które są zgodne ze strukturą MQDLH . Dane zwykle pochodzą z oryginalnego komunikatu. Nie ma on zastosowania do danych liczbowych w samej strukturze MQDLH .

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

DLFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych, które są następujące MQDLH.

Określa nazwę formatu danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu).

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak reguły dla pola MDFMT w produkcie MQMD.

Długość tego pola jest podawana przez produkt LNFMT. Wartością początkową tego pola jest FMNONE.

DLPAN (28-bajtowy łańcuch znaków)

Nazwa aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format nazwy zależy od pola DLPAT . Zapoznaj się z opisem pola MDPAN w podręczniku [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Jeśli jest to menedżer kolejek, który przekierował komunikat do kolejki niedostarczonych komunikatów, DLPAN zawiera pierwsze 28 znaków nazwy menedżera kolejek. Jeśli to konieczne, nazwa jest dopełniona odstępami.

Długość tego pola jest podawana przez produkt LNPAN. Początkowa wartość tego pola to 28 znaków odstępów.

DLPAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

To pole ma takie samo znaczenie, jak pole MDPAT w deskrytorze komunikatu MQMD (szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124).

Jeśli jest to menedżer kolejek, który przekierowuje komunikat do kolejki niedostarczonych komunikatów, parametr DLPAT ma wartość ATQM.

Wartością początkową tego pola jest 0.

DLPD (8-bajtowy łańcuch znaków)

Data umieszczenia komunikatu w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format używany dla daty, w której to pole jest generowane przez menedżer kolejek:

- YYYYMMDD

gdzie znaki reprezentują:

YYYY

rok (cztery cyfry)

MM

miesiąc w roku (01 do 12)

DD

Dzień miesiąca (01 do 31)

Czas Greenwich (Greenwich Mean Time) jest używany dla pól DLPD i DLPT , pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Długość tego pola jest podawana przez produkt LNPDAT. Początkowa wartość tego pola to osiem znaków odstępu.

DLPT (8-bajtowy łańcuch znaków)

Czas umieszczenia komunikatu w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format używany w czasie, gdy pole jest generowane przez menedżer kolejek:

- HHMMSSSTH

gdzie znaki reprezentują (w kolejności):

HH

godzin (od 00 do 23)

MM

minuty (od 00 do 59)

SS

sekundy (od 00 do 59; patrz uwaga później w tym temacie)

T

Dziesiątych sekundy (od 0 do 9)

H

Setnych części sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z dokładnym standardem czasu, możliwe jest zwrócenie wartości 60 lub 61 w ciągu sekundy w DLPT. Dodatkowa sekunda występuje wtedy, gdy sekundy przestępne są wstawiane w globalnym standardzie czasu.

Czas Greenwich (Greenwich Mean Time) jest używany dla pól DLPD i DLPT, pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Długość tego pola jest podawana przez produkt LNPTIM. Początkowa wartość tego pola to osiem znaków odstępu.

DLREA (dziesięciocyfrowa liczba całkowita ze znakiem)

Komunikat przyczyny dotarł do kolejki niedostarczonych komunikatów (niedostarczonych komunikatów).

Określa przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów, a nie w oryginalnej kolejce docelowej. Musi to być jedna z wartości FB* lub RC* (na przykład RC2053). Aby uzyskać szczegółowe informacje na temat wspólnych wartości FB*, które mogą wystąpić, należy zapoznać się z opisem w polu *MDFB* w publikacji [“MQMD-deskryptor komunikatu” na stronie 1124](#).

Jeśli wartość mieści się w zakresie od FBIFST do FBILST, rzeczywisty kod błędu IMS może być określony przez odjęcie FBIERR od wartości pola *DLREA*.

Niektóre wartości FB* występują tylko w tym polu. Odnoszą się one do komunikatów repozytorium, komunikatów wyzwalacza lub komunikatów kolejki transmisji, które są przesyłane do kolejki niedostarczonych komunikatów. Są to następujące wartości:

FBABEG

Nie można uruchomić aplikacji.

Aplikacja przetwarzający komunikat wyzwalacza nie mogła uruchomić aplikacji o nazwie określonej w polu TMAI komunikatu wyzwalacza. Patrz [“MQTM-komunikat wyzwalacza” na stronie 1255](#).

FBATYP

Błąd typu aplikacji.

Aplikacja przetwarzający komunikat wyzwalacza nie mogła uruchomić aplikacji, ponieważ pole TMAI komunikatu wyzwalacza nie jest poprawne. Patrz [“MQTM-komunikat wyzwalacza” na stronie 1255](#).

FBB OCD

Usunięto kanał odbierający klastry.

Komunikat był przeznaczony dla kolejki transmisji klastra przeznaczonej dla kolejki klastra, która została otwarta z opcją FBIERR . Zdalny kanał odbierający klastry, który ma zostać użyty do przesłania komunikatu do kolejki docelowej, został usunięty przed wystaniem komunikatu. Ponieważ określono parametr FBIERR , do przesłania komunikatu można użyć tylko kanału wybranego, gdy kolejka została otwarta. Ponieważ ten kanał nie jest już dostępny, komunikat został umieszczony w kolejce niedostarczonych komunikatów.

FBN ARM

Komunikat nie jest komunikatem repozytorium.

FBS BCX

Komunikat został zatrzymany przez wyjście automatycznej definicji kanału.

FBS BMX

Komunikat został zatrzymany przez wyjście komunikatów kanału.

FBT M

Struktura MQTM nie jest poprawna lub nie istnieje.

Pole MDFMT w MQMD określa FMTM, ale komunikat nie rozpoczyna się od poprawnej struktury MQTM . Na przykład: mnemonik *TMSID* może nie być poprawny. Możliwe, że *TMVER* nie zostanie rozpoznany. Długość komunikatu wyzwalacza może być niewystarczająca, aby pomieścić strukturę MQTM .

FBX QME

Komunikat w kolejce transmisji nie jest poprawny.

Agent kanału komunikatów stwierdził, że komunikat w kolejce transmisji nie jest w poprawnym formacie. Agent kanału komunikatów umieszcza komunikat w kolejce niedostarczonych komunikatów przy użyciu tego kodu informacji zwrotnych.

Wartością początkową tego pola jest RCNONE.

DLSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DLSIDV

Identyfikator struktury nagłówka niedostarczonych komunikatów.

Wartością początkową tego pola jest DLSIDV.

DLVER (dziesięciocyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DLVER1

Numer wersji dla struktury nagłówka niedostarczonych komunikatów.

Następująca stała określa numer wersji bieżącej wersji:

DLVERC

Bieżąca wersja struktury nagłówka niedostarczonych komunikatów.

Wartością początkową tego pola jest DLVER1.

Wartości początkowe

Tabela 138. Wartości początkowe pól w produkcie MQDLH.

Wyświetla listę nazw stałych MQDLH i ich wartości.

Nazwa pola	Nazwa stałej	Wartość stałej
ID_DLSID	DLSIDV	'DLH~'
NIGDY	DLVER1	1
DLREA	RCNONE	0
DLDQ	Brak	Puste
DLDM	Brak	Puste
DLENC	Brak	0
DLCSI	CSUNDF	0
DLFMT	FMNONE	Puste
DLPAT	Brak	0
DLPAN	Brak	Puste
DLPD	Brak	Puste
DLPT	Brak	Puste

Uwagi:

1. Symbol ~ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQDLH Structure
D*
D* Structure identifier
D DLSID 1 4 INZ('DLH ')
D* Structure version number
D DLVER 5 8I 0 INZ(1)
D* Reason message arrived on dead-letter(undelivered-message) queue
D DLREA 9 12I 0 INZ(0)
D* Name of original destination queue
D DLDQ 13 60 INZ
D* Name of original destination queue manager
D DLDM 61 108 INZ
D* Numeric encoding of data that follows MQDLH
D DLENC 109 112I 0 INZ(0)
D* Character set identifier of data that follows MQDLH
D DLCSI 113 116I 0 INZ(0)
D* Format name of data that follows MQDLH
D DLFMT 117 124 INZ(' ')
D* Type of application that put message on dead-letter
D* (undelivered-message) queue
D DLPAT 125 128I 0 INZ(0)
D* Name of application that put message on dead-letter
D* (undelivered-message) queue
D DLPAN 129 156 INZ
D* Date when message was put on dead-letter (undelivered-message) queue
D DLPD 157 164 INZ
D* Time when message was put on the dead-letter (undelivered-message) queue
D DLPT 165 172 INZ

```

MQDMHO-usuwanie opcji uchwytu komunikatu

Struktura **MQDMHO** umożliwia aplikacjom określanie opcji, które sterują sposobem usuwania uchwytów komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym w wywołaniu **MQDLTMH**.

Zestaw znaków i kodowanie: Dane w programie **MQDMHO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1084](#)
- [“Wartości początkowe” na stronie 1084](#)
- [“Deklaracja RPG” na stronie 1085](#)

Pola

Struktura **MQDMHO** zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

DMOPT (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

DMNONE

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **DMNONE**.

DMSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

DMSIDV

Identyfikator struktury opcji uchwytu komunikatu usuwania.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **DMSIDV**.

DMVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

DMVER1

Version-1 -usuń strukturę opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

DMVERC

Bieżąca wersja struktury opcji uchwytu komunikatu usuwania.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **DMVER1**.

Wartości początkowe

Tabela 139. Początkowe wartości pól w MQDMHO		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>DMSID</i>	DMSIDV	' DMHO '
<i>DMVER</i>	DMVER1	1
<i>DMOPT</i>	DMNONE	0

Deklaracja RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID              1      4      INZ('DMHO')
D*
D* Structure version number
D  DMVER              5      8I 0  INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT              9      12I 0 INZ(0)
```

MQDMPO-opcje usuwania właściwości komunikatu

Struktura definiująca opcje właściwości usuwania komunikatu.

Przegląd

Przeznaczenie: Struktura MQDMPO umożliwia aplikacjom określanie opcji sterujących sposobem usuwania właściwości komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQDLTMP.

Zestaw znaków i kodowanie: Dane w tabeli MQDMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1085](#)
- [“Wartości początkowe” na stronie 1086](#)
- [“Deklaracja RPG” na stronie 1086](#)

Pola

Struktura MQDMPO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

DPOPT (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPOPT.

Opcje lokalizacji: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości.

DPDELF

Usuwa pierwszą właściwość, która jest zgodna z podaną nazwą.

DPDELC

Usuwa właściwość wskazywaną przez kursor właściwości. Jest to właściwość, która została ostatnio sprawdzona przy użyciu opcji IPINQF lub IPINQN.

Kursor właściwości zostanie zresetowany, gdy uchwyt komunikatu zostanie ponownie wykorzystany. Jest ona również resetowana, gdy uchwyt komunikatu jest określony w polu *HMSG* w tabeli MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany, lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w strukturze MQGET w strukturze wywołania MQGET lub MQPMO w wywołaniu MQPUT.

Wywołanie nie powiodło się z kodem zakończenia CCFAIL i przyczyną RC2471, jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony. Nie powiedzie się również z tymi kodami, jeśli właściwość wskazywana przez kursor właściwości została już usunięta.

Jeśli żadna z tych opcji nie jest wymagana, można użyć następującej opcji:

DPNONE

Nie określono żadnych opcji.

Wartością początkową tego pola wejściowego jest DPDELF.

DPSID (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPSID.

Jest to identyfikator struktury. Wartość musi być następująca:

DPSIDV

Identyfikator struktury opcji usuwania właściwości komunikatu.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest DPSIDV.

DPVER (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPVER.

Jest to numer wersji struktury. Wartość musi być następująca:

DPVER1

Numer wersji dla struktury opcji usuwania właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

DPVERC

Bieżąca wersja struktury opcji usuwania właściwości komunikatu.

To pole jest zawsze polem wejściowym. Początkowa wartość tego pola to DPVER1.

Wartości początkowe

Tabela 140. Początkowe wartości pól w MQDPMO		
Nazwa pola	Nazwa stałej	Wartość stałej
DPSID	DPSIDV	' DMP0 '
DPVER	DPVER1	1
DPOPT	Opcje sterujące działaniem komendy MQDLTMP	DPNONE

Deklaracja RPG

```
D* MQDPMO Structure
D*
D*
D* Structure identifier
D  DPSID              1      4    INZ(' DMP0 ')
D*
D* Structure version number
D  DPVER              5      8I 0  INZ(1)
D*
** Options that control the action of
D* MQDLTMP
D  DPOPT              9     12I 0  INZ(0)
```

MQEPH-osadzony nagłówek PCF

Przegląd

Przeznaczenie

Struktura MQEPH opisuje dodatkowe dane, które są obecne w komunikacie, gdy jest to komunikat PCF (programmable command format). Pole *EPPFH* definiuje parametry PCF, które są zgodne z tą strukturą, co pozwala na śledzenie danych komunikatu PCF z innymi nagłówkami.

Nazwa formatu

EPFMT

Zestaw znaków i kodowanie

Dane w tabeli MQEPH muszą znajdować się w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Dane te są podawane przez atrybut menedżera kolejek produktu *CCSID*.

Ustaw zestaw znaków i kodowanie wartości MQEPH w polach *MDCSI* i *MDENC* w:

- MQMD (jeśli struktura MQEPH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQEPH (wszystkie inne obserwacje).

Użycie

Struktury MQEPH nie można używać do wysyłania komend do serwera komend ani do żadnego innego serwera akceptującego menedżera kolejek PCF.

Podobnie serwer komend lub inny serwer akceptujący menedżera kolejek PCF nie generują odpowiedzi ani zdarzeń zawierających struktury MQEPH.

- [“Pola” na stronie 1087](#)
- [“Wartości początkowe” na stronie 1089](#)
- [“Deklaracja RPG” na stronie 1089](#)

Pola

Struktura MQEPH zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**:

EPCSI (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator zestawu znaków danych, który jest zgodny ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma on zastosowania do danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest EPCUND.

EPENC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kodowanie liczbowe dla danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie dotyczy to danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest 0.

EPFLG (10-cyfrowa liczba całkowita ze znakiem)

Dozwolone są następujące wartości:

EPNONE

Nie określono żadnych flag. *MDCSI* EPNONE jest zdefiniowana w dokumentacji programu pomocowego. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

EPCSEM

Zestaw znaków parametrów zawierających dane znakowe jest określany indywidualnie w obrębie pola *CCSID* w każdej strukturze. Zestaw znaków w polach *EPSID* i *EPFMT* jest zdefiniowany przez *CCSID* w strukturze nagłówka poprzedzający strukturę MQEPH lub przez pole *MDCSI* w strukturze MQMD, jeśli wartość MQEPH jest na początku komunikatu.

Wartością początkową tego pola jest EPNONE.

EPFMT (8-bajtowy łańcuch znaków)

Jest to nazwa formatu danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF.

Wartością początkową tego pola jest EPFMNO.

EPLEN (10-cyfrowa liczba całkowita ze znakiem)

Jest to ilość danych poprzedzająca następną strukturę nagłówka. Obejmuje ona:

- Długość nagłówka MQEPH
- Długość wszystkich parametrów PCF następujących po nagłówku
- Każde puste dopełnienie po tych parametrach

Wartość EPLEN musi być wielokrotnością 4.

Część o stałej długości struktury jest zdefiniowana przez EPSTLF.

Wartością początkową tego pola jest 68.

EPPCFH (MQCFH)

Jest to nagłówek programowalnego formatu komend (PCF), definiujący parametry PCF, które są zgodne ze strukturą MQEPH. Dzięki temu można śledzić dane komunikatu PCF z innymi nagłówkami.

Nagłówek PCF jest początkowo zdefiniowany z następującymi wartościami:

<i>Tabela 141. Początkowe wartości pól w EPPCFH</i>		
Nazwa pola	Nazwa stałej	Wartość stałej
EP3TYP	CFTNON	0
EP3LEN	FHLENV	36
EP3VER	FHVER3	3
EP3CMD	CMNONE	0
EP3SEQ	Brak	1
EP3CTL	CFCLST	1
EEP3CC	CCOK	0
EP3REA	RCBRAK	0
EP3CNT	Brak	0

Aplikacja musi zmienić wartość EP3TYP z CFTNON na poprawny typ struktury dla użycia, który jest używany w osadzonym nagłówku PCF.

EPSID (4-bajtowy łańcuch znaków)

Wartość musi być następująca:

ID EPSTID

Identyfikator struktury nagłówka Embedded PCF.

Wartością początkową tego pola jest EPSTID.

EPVER (10-cyfrowa liczba całkowita ze znakiem)

Możliwe wartości:

EPVER1

Numer wersji dla osadzonej struktury nagłówka PCF.

Następująca stała określa numer wersji bieżącej wersji:

EPVER3

Bieżąca wersja wbudowanej struktury nagłówka PCF.

Początkowa wartość tego pola to EPVER3.

Wartości początkowe

Tabela 142. Początkowe wartości pól w tabeli MQEPH		
Nazwa pola	Nazwa stałej	Wartość stałej
EPSID	ID EPSTID	'EP- -'
EPVER	EPVER1	1
EPLEN	EPSTLF	68
EPENC	Brak	0
EPCSI	EPCUND	0
EPFMT	EPFMNO	Puste
EPFLG	EPNONE	0
EPPCFH	Nazwy i wartości zdefiniowane w produkcie Tabela 141 na stronie 1088	0

Uwaga:

1. Symbol – reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQEPH Structure
D*
D* Structure identifier
D EPSID          1          4
D* Structure version number
D EPVER          5          8I 0
D* Total length of MQEPH including MQCFHand parameter structures
D* that follow
D EPLEN          9          12I 0
D* Numeric encoding of data that follows last PCF parameter structure
D EPENC          13         16I 0
D* Character set identifier of data that follows last PCF parameter
D* structure
D EPCSI          17         20I 0
D* Format name of data that follows last PCF parameter structure
D EPFMT          21         28
D* Flags
D EPFLG          29         32I 0
D* Programmable Command Format Header
D EP3TYP          33         36I 0
D EP3LEN          37         40I 0
D EP3VER          41         44I 0
D EP3CMD          45         48I 0
D EP3SEQ          49         52I 0
D EP3CTL          53         56I 0
D EP3CC           57         60I 0
D EP3REA          61         64I 0
D EP3CNT          65         68I 0

```

MQGMO-Opcje Get-message

Struktura MQGMO umożliwia aplikacji określanie opcji, które sterują sposobem usuwania komunikatów z kolejek.

Przegląd

Przeznaczenie

Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQGET.

Wersja

Bieżąca wersja produktu MQGMO to GMVER4. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję produktu MQGMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *GMVER* ustawioną na wartość GMVER1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *GMVER* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie

Dane w produkcie MQGMO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu *CodedCharSetId* oraz kodowanie lokalnego menedżera kolejek nadawanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1090](#)
- [“Wartości początkowe” na stronie 1111](#)
- [“Deklaracja RPG” na stronie 1111](#)

Pola

Struktura MQGMO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

GMGST (1 bajtowy łańcuch znaków)

Flaga wskazująca, czy wczytany komunikat znajduje się w grupie.

Ma jedną z następujących wartości:

GSNIG

Komunikat nie znajduje się w grupie.

GSMIG

Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

GSLMIG

Komunikat jest ostatnim w grupie.

Ta wartość jest również wartością zwracaną, jeśli grupa składa się tylko z jednego komunikatu.

To pole jest polem wyjściowym. Wartością początkową tego pola jest GSNIG. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż GMVER2.

GMMH (10-cyfrowa liczba całkowita ze znakiem)

uchwyt komunikatu

Jeśli określono opcję GMPRAQ, a atrybut kolejki PRPCTL nie jest ustawiony na PRPRFH, to jest to uchwyt do komunikatu, który jest zapełniany właściwościami komunikatu pobieranego z kolejki. Uchwyt jest tworzony za pomocą wywołania MQCRTMH. Wszystkie właściwości, które są już powiązane z uchwytem, są czyszczone przed pobraniem komunikatu.

Można również określić następującą wartość:

MQHM_NONE

Nie podano uchwytu komunikatu.

Żaden deskryptor komunikatu nie jest wymagany w wywołaniu MQGET, jeśli poprawny uchwyt komunikatu jest dostarczany i używany na wyjściu w celu zawierania właściwości komunikatu, deskryptor komunikatu powiązany z uchwytem komunikatu jest używany dla pól wejściowych.

Jeśli deskryptor komunikatu jest określony w wywołaniu MQGET, zawsze ma on pierwszeństwo przed deskryptorem komunikatu powiązonym z uchwytem komunikatu.

Jeśli określono parametr GMPRRF lub określono parametr GMPRAQ, a atrybut kolejki PRPCTL ma wartość PRPRFH, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2026, jeśli nie określono parametru deskryptora komunikatu.

W przypadku powrotu z wywołania MQGET właściwości i deskryptor komunikatu powiązane z tym uchwycem komunikatu są aktualizowane w celu odzwierciedlenia stanu pobranego komunikatu (a także deskryptora komunikatu, jeśli został on dostarczony w wywołaniu MQGET). Właściwości komunikatu można następnie dowiedzieć się za pomocą wywołania MQINQMP.

Poza rozszerzeniami deskryptora komunikatu, jeśli istnieje, właściwość, która może zostać zapytana za pomocą wywołania MQINQMP, nie jest zawarta w danych komunikatu. Jeśli komunikat w kolejce zawiera właściwości w danych komunikatu, są one usuwane z danych komunikatu przed zwróceniem danych do aplikacji.

Jeśli żaden uchwyt komunikatu nie jest podany lub wersja jest mniejsza niż wartość GMVER4, należy podać poprawny deskryptor komunikatu w wywołaniu MQGET. Wszystkie właściwości komunikatu (z wyjątkiem właściwości zawartych w deskrypcie komunikatu) są zwracane w danych komunikatu z uwzględnieniem wartości opcji właściwości w strukturze MQGMO oraz w atrybucie kolejki PRPCTL.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest HMNONE. To pole jest ignorowane, jeśli parametr GMVER jest mniejszy niż GMVER4.

GMMO (10-cyfrowa liczba całkowita ze znakiem)

Opcje kontrolujące kryteria wyboru używane dla komendy MQGET.

Te opcje pozwalają aplikacji wybrać, które pola w parametrze MSGDSC są używane do wybierania komunikatu zwracanego przez wywołanie MQGET. Aplikacja ustawia wymagane opcje w tym polu, a następnie ustawia odpowiednie pola w parametrze MSGDSC na wartości wymagane dla tych pól. Tylko komunikaty, które mają te wartości w strukturze MQMD dla komunikatu, są kandydatami do pobrania przy użyciu tego parametru MSGDSC w wywołaniu MQGET. Pola, dla których odpowiednia opcja zgodności nie została określona, są ignorowane podczas wybierania komunikatu do zwrócenia. Jeśli w wywołaniu MQGET nie będą używane żadne kryteria wyboru (oznacza to, że dowolny komunikat jest akceptowalny), parametr GMMO powinien zostać ustawiony na wartość MONONE.

Jeśli określono wartość GMLOGO, tylko niektóre komunikaty są zakwalifikowane do powrotu przez następne wywołanie MQGET:

- Jeśli nie istnieje żadna bieżąca grupa lub komunikat logiczny, do zwrotu kwalifikują się tylko komunikaty, które mają MDSEQ równe 1 i MDOFF równe 0. W takiej sytuacji można wybrać jedną lub więcej spośród następujących opcji, aby wybrać, które z uprawnionych komunikatów są zwracane:
 - MOMSGI
 - MOCORI
 - MOGRPI
- Jeśli istnieje bieżąca grupa lub komunikat logiczny, do zwrotu kwalifikuje się tylko następny komunikat w grupie lub następny segment w komunikacie logicznym, który nie może zostać zmieniony przez określenie opcji MO*.

W obu przypadkach nadal można określić opcje zgodności, które nie mają zastosowania, ale wartość odpowiedniego pola w parametrze MSGDSC musi być zgodna z wartością odpowiedniego pola w komunikacie, które ma zostać zwrócone; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2247 jest to warunek, który nie jest spełniony.

Wartość GMMO jest ignorowana, jeśli określono wartość GMMUC lub GMBRWC.

Można określić jedną lub więcej spośród następujących opcji:

MOMSGI

Pobranie komunikatu z określonym identyfikatorem komunikatu.

Ta opcja określa, że komunikat, który ma zostać pobrany, musi mieć identyfikator komunikatu, który jest zgodny z wartością pola MDMID w parametrze MSGDSC wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowaniami, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja nie zostanie podana, pole MDMID w parametrze MSGDSC zostanie zignorowane, a każdy identyfikator komunikatu zostanie dopasowany.

Uwaga: Identyfikator komunikatu MINONE jest specjalną wartością, która jest zgodna z dowolnym identyfikatorem komunikatu w strukturze MQMD dla komunikatu. Dlatego określenie parametru MOMSGI z parametrem MINONE jest takie samo, jak nie określenie parametru MOMSGI.

MOCORI

Pobieranie komunikatu z określonym identyfikatorem korelacji.

Ta opcja określa, że komunikat, który ma zostać pobrany, musi mieć identyfikator korelacji, który jest zgodny z wartością pola *MDCID* w parametrze *MSGDSC* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład identyfikator komunikatu).

Jeśli ta opcja nie zostanie podana, pole *MDCID* w parametrze *MSGDSC* zostanie zignorowane, a dowolny identyfikator korelacji zostanie dopasowany.

Uwaga: Identyfikator korelacji CINONE jest specjalną wartością, która jest zgodna z dowolnym identyfikatorem korelacji w strukturze MQMD dla komunikatu. Dlatego określenie MOCORI z CINONE jest takie samo, jak nie określenie MOCORI.

MOGRPI

Pobieranie komunikatu z określonym identyfikatorem grupy.

Ta opcja określa, że komunikat, który ma zostać pobrany, musi mieć identyfikator grupy, który jest zgodny z wartością pola *MDGID* w parametrze *MSGDSC* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja nie zostanie podana, pole *MDGID* w parametrze *MSGDSC* zostanie zignorowane, a każdy identyfikator grupy zostanie dopasowany.

Uwaga: Identyfikator grupy GINONE jest specjalną wartością, która jest zgodna z dowolnym identyfikatorem grupy w strukturze MQMD komunikatu. Dlatego podanie parametru MOGRPI z parametrem GINONE jest takie samo, jak nie określenie parametru MOGRPI.

MOSEQN

Pobieranie komunikatu z określonym numerem kolejnym komunikatu.

Ta opcja określa, że komunikat, który ma zostać pobrany, musi mieć numer kolejny komunikatu, który jest zgodny z wartością pola *MDSEQ* w parametrze *MSGDSC* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład identyfikator grupy).

Jeśli ta opcja nie zostanie podana, pole *MDSEQ* w parametrze *MSGDSC* zostanie zignorowane, a wszystkie kolejne numery kolejnego komunikatu będą zgodne.

MOOFFS

Pobranie komunikatu z określonym przesunięciem.

Ta opcja określa, że komunikat, który ma zostać pobrany, musi mieć przesunięcie zgodne z wartością pola *MDOFF* w parametrze *MSGDSC* wywołania MQGET. Ta zgodność jest dodatkowo zgodna z innymi dopasowań, które mogą mieć zastosowanie (na przykład numer kolejny komunikatu).

Jeśli ta opcja nie zostanie podana, pole *MDOFF* w parametrze *MSGDSC* zostanie zignorowane, a wszystkie przesunięcia zostaną dopasowane.

Jeśli żadna z opisanych opcji nie jest określona, można użyć następującej opcji:

MONONE

Brak dopasowań.

Ta opcja określa, że nie będą używane żadne zgodne elementy w wyborze komunikatu, który ma zostać zwrócony. W związku z tym wszystkie komunikaty znajdujące się w kolejce są zakwalifikowane do pobrania (ale podlegają kontroli za pomocą opcji GMAMSA, GMASGA i GMCMPM).

MONONE jest zdefiniowana w dokumentacji programu pomocy. Nie jest zamierzone, aby ta opcja była używana z żadną inną opcją MO*, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

To pole jest polem wejściowym. Wartością początkową tego pola jest MOMSGI z MOCORI. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż *GMVER2*.

Uwaga: Początkowa wartość pola *GMMO* jest zdefiniowana pod kątem zgodności z menedżerami kolejek wcześniejszych wersji. Jednak podczas odczytywania serii komunikatów z kolejki bez użycia kryteriów wyboru, ta wartość początkowa wymaga, aby aplikacja zresetowała pola *MDMID* i *MDCID* do MINONE i CINONE przed każdym wywołaniem MQGET. Aby uniknąć konieczności resetowania *MDMID* i *MDCID*, należy ustawić parametr *GMVER* na wartość *GMVER2*, a *GMMO* na MONONE.

GMOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje, które sterują działaniem MQGET.

Można podać zero lub więcej następujących opcji opisanych poniżej. Jeśli wymagane jest dodanie więcej niż jednego, wartości można dodać (nie należy dodawać tej samej stałej więcej niż raz). Kombinacje opcji, które nie są poprawne, są oznaczane; wszystkie pozostałe kombinacje są poprawne.

Opcje oczekiwania: Następujące opcje odnoszą się do oczekiwania na przybycie komunikatów do kolejki:

GMWT

Poczekaj na przybycie komunikatu.

Aplikacja ma czekać, aż pojawi się odpowiedni komunikat. Maksymalny czas oczekiwania aplikacji jest określony w *GMWT*.

Jeśli żądania MQGET są zablokowane lub żądania MQGET zostają zablokowane podczas oczekiwania, oczekiwanie zostanie anulowane, a wywołanie zakończy się z kodem CCFAIL i kodem przyczyny RC2016, niezależnie od tego, czy w kolejce znajdują się odpowiednie komunikaty.

Ta opcja może być używana z opcjami GMBRWF lub GMBRWN.

Jeśli kilka aplikacji oczekuje w tej samej współużytkowanej kolejce, aplikacja lub aplikacje, które są aktywowane po nadejściu odpowiedniego komunikatu, zostaną opisane w dalszej części tej sekcji.

Uwaga: W poniższym opisie znajduje się wywołanie przeglądania MQGET, które określa jedną z opcji przeglądania, ale nie dla parametru GMLK; wywołanie MQGET określające opcję GMLK jest traktowane jako wywołanie bez przeglądania.

- Jeśli trwa oczekiwanie co najmniej jednej nieprzeglądających wywołań MQGET, ale nie oczekuje na przeglądanie wywołań MQGET, jeden jest aktywowany.
- Jeśli trwa oczekiwanie co najmniej jednej operacji przeglądania MQGET, ale nie oczekuje na przeglądanie wywołań MQGET, wszystkie te wywołania są aktywowane.
- Jeśli co najmniej jedno wywołanie MQGET bez przeglądania i jedno lub więcej wywołań przeglądania MQGET jest oczekujących, zostanie aktywowane jedno wywołanie MQGET bez przeglądania, a także brak, niektóre lub wszystkie wywołania MQGET. (Nie można przewidzieć liczby aktywnych wywołań MQGET, ponieważ zależy ona od uwarunkowań planowania systemu operacyjnego i innych czynników).

Jeśli w tej samej kolejce oczekuje więcej niż jedno wywołanie funkcji MQGET, tylko jedna zostanie aktywowana. W takiej sytuacji menedżer kolejek próbuje nadać priorytet oczekujemu wywołaniu, które nie będą przeglądać w następującej kolejności:

1. Konkretnie żądania get-wait, które mogą być spełnione tylko przez niektóre komunikaty, na przykład takie, które mają konkretną wartość *MDMID* lub *MDCID* (lub obie te wartości).
2. Ogólne żądania pobierania, które mogą być spełnione przez dowolny komunikat.

Należy zwrócić uwagę na następujące kwestie:

- W ramach pierwszej kategorii nie nadano dodatkowego priorytetu dla bardziej konkretnych żądań pobierania czasu oczekiwania, na przykład tych, które określają zarówno produkt *MDMID*, jak i *MDCID*.
- W ramach jednej z kategorii nie można przewidzieć, która aplikacja jest wybrana. W szczególności, najdłuższy czas oczekiwania aplikacji nie musi być wybrany.
- Długość ścieżki oraz uwagi dotyczące planowania priorytetów w systemie operacyjnym mogą oznaczać, że aplikacja oczekująca o niższym priorytecie systemu operacyjnego niż oczekiwano pobiera komunikat.
- Może się również zdarzyć, że aplikacja, która nie oczekuje na pobranie komunikatu, będzie preferowana względem tego, który jest.

GMWT jest ignorowane, jeśli jest określony przy użyciu GMBRWC lub GMMUC; nie jest zgłaszany żaden błąd.

GMNWT

Zwróć natychmiast, jeśli nie ma odpowiedniego komunikatu.

Aplikacja nie może czekać, jeśli nie jest dostępny odpowiedni komunikat. Jest to przeciwieństwo opcji GMWT i jest ona zdefiniowana w dokumentacji programu pomocowego. Jest to wartość domyślna, jeśli nie zostanie podana żadna wartość.

GMFIQ

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

Ta opcja wymusza niepowodzenie wywołania MQGET, jeśli menedżer kolejek znajduje się w stanie wygaszania.

Jeśli ta opcja jest określona razem z GMWT, a oczekiwanie jest zaległe w momencie, w którym menedżer kolejek przechodzi do stanu wygaszania:

- Oczekiwanie zostało anulowane, a wywołanie zwraca kod zakończenia CCFAIL z kodem przyczyny RC2161.

Jeśli wartość GMFIQ nie zostanie określona, a menedżer kolejek przejdzie do stanu wygaszania, to oczekiwanie nie zostanie anulowane.

Opcje punktu synchronizacji: Następujące opcje odnoszą się do udziału wywołania MQGET w ramach jednostki pracy:

GMSYP

Pobierz komunikat z elementem sterującym punktu synchronizacji.

Żądanie ma działać w ramach normalnych protokołów jednostkowych pracy. Komunikat jest oznaczony jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzana. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy jest wycofana.

Jeśli ta opcja lub wartość GMNSYP nie zostanie podana, żądanie pobrania nie znajduje się w obrębie jednostki pracy.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF
- GMBRWC
- GMBRWN
- GMLK
- GMNSYP
- GMPSYP
- GMUNLK

GMPSYP

Pobierz komunikat z elementem sterującym punktu synchronizacji, jeśli komunikat jest trwały.

Żądanie ma działać w normalnych protokołach jednostki pracy, ale tylko wtedy, gdy pobrany komunikat jest trwały. A persistent message has the value PEPER in the *MDPER* field in MQMD.

- Jeśli komunikat jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja została określona przez aplikację GMSYP.
- Jeśli komunikat nie jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja określiła wartość GMNSYP (szczegółowe informacje można znaleźć w poniższej sekcji).

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF
- GMBRWC
- GMBRWN
- GMCMPM
- GMNSYP
- GMSYP
- GMUNLK

GMNSYP

Pobierz komunikat bez elementu sterującego punktu synchronizacji.

Wniosek ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest natychmiast usuwany z kolejki (o ile nie jest to żądanie przeglądania). Komunikat nie może zostać ponownie udostępniony przez utworzenie kopii zapasowej jednostki pracy.

Ta opcja jest przyjmowana, jeśli określono parametr GMBRWF lub GMBRWN.

Jeśli ta opcja i GMSYP nie zostaną określone, żądanie pobrania nie znajduje się w obrębie jednostki pracy.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMSYP
- GMPSYP

Opcje przeglądania: Następujące opcje są powiązane z przeglądaniem komunikatów w kolejce:

GMBRWF

Odszukaj od początku kolejki.

Gdy kolejka jest otwierana za pomocą opcji OOBROW, kursor przeglądania jest ustanawiany, umieszczany logicznie przed pierwszym komunikatem w kolejce. Kolejne wywołania MQGET określające opcję GMBRWF, GMBRWN lub GMBRWC mogą być używane do pobierania komunikatów z kolejki nieniszczących. Kursor przeglądania oznacza pozycję w obrębie komunikatów w kolejce, z której następne wywołanie MQGET z wyszukiwaniem GMBRWN dla odpowiedniego komunikatu.

Wywołanie MQGET z GMBRWF powoduje, że poprzednia pozycja kursora przeglądania zostanie zignorowana. Wczytany jest pierwszy komunikat w kolejce, który spełnia warunki określone w deskrypcji komunikatu. Komunikat pozostaje w kolejce, a kursor przeglądania znajduje się w tym komunikacie.

Po wywołaniu tej operacji kursor przeglądania jest ustawiony na zwróconej wiadomości. Jeśli komunikat zostanie usunięty z kolejki przed wywołaniem następnej wywołania MQGET z wartością GMBRWN, kursor przeglądania pozostaje w kolejce zajmowanej przez komunikat, nawet jeśli pozycja ta jest teraz pusta.

Opcja GMMUC może być następnie używana z nieprzeglądaniem wywołania MQGET, jeśli jest to wymagane, w celu usunięcia komunikatu z kolejki.

Kursor przeglądania nie jest przenoszony za pomocą wywołania MQGET bez przeglądania przy użyciu tego samego uchwytu *HOBJ*. Nie jest też przenoszona przez przeglądanie wywołania MQGET, które zwraca kod zakończenia CCFAIL lub kod przyczyny RC2080.

Opcja GMLK może zostać określona razem z tą opcją, aby spowodować zablokowanie komunikatu, który ma zostać zablokowany.

Parametr GMBRWF może być określony z dowolną poprawną kombinacją opcji GM* i MO*, które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli określono wartość GMLOGO, komunikaty są przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty są przeglądane w porządku fizycznym. Jeśli określono parametr GMBRWF, można przełączać się między porządkiem logicznym a porządkiem fizycznym, ale kolejne wywołania MQGET używające komendy GMBRWN muszą przeglądać kolejkę w tej samej kolejności, co ostatnie wywołanie, które określił GMBRWF dla uchwytu kolejki.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które usuwają komunikaty z kolejki. Jeśli określono parametr GMBRWF, menedżer kolejek ignoruje informacje o grupie i segmencie w celu przeglądania, a następnie skanuje kolejkę, tak jakby nie było żadnej bieżącej grupy i nie ma bieżącego komunikatu logicznego. Jeśli wywołanie metody theMQGET powiedzie się (kod zakończenia CCOK lub CCWARN), informacje o grupach i segmentach służące do przeglądania są ustawione na wartość zwróconego komunikatu. Jeśli wywołanie nie powiedzie się, informacje o grupach i segmentach pozostaną takie same, jak przed wywołaniem.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMBRWN

Przeglądaj z bieżącej pozycji w kolejce.

Kursor przeglądania jest zaawansowany do następnego komunikatu w kolejce, który spełnia kryteria wyboru określone w wywołaniu MQGET. Komunikat jest zwracany do aplikacji, ale pozostaje w kolejce.

Po otwarciu kolejki dla przeglądania pierwsze wywołanie przeglądania za pomocą uchwytu ma ten sam efekt, niezależnie od tego, czy określa opcję GMBRWF, czy GMBRWN.

Jeśli komunikat zostanie usunięty z kolejki przed wywołaniem następnej wywołania MQGET z wartością GMBRWN, kursor przeglądania pozostaje logicznie na pozycji w kolejce zajmowanej przez komunikat, nawet jeśli pozycja ta jest teraz pusta.

Komunikaty są zapisywane w kolejce na jeden z dwóch sposobów:

- FIFO w ramach priorytetu (MSPRIO), lub
- FIFO niezależnie od priorytetu (MSFIFO)

Atrybut kolejki *MsgDeliverySequence* wskazuje, która metoda ma zastosowanie (szczegółowe informacje na ten temat zawiera sekcja [“Atrybuty dla kolejek”](#) na stronie 1395).

Jeśli w kolejce znajduje się *MsgDeliverySequence* MSPRIO, a komunikat pojawia się w kolejce o wyższym priorytecie niż ten aktualnie wskazywany na kursor przeglądania, to komunikat ten nie zostanie znaleziony podczas bieżącego przeglądania kolejki przy użyciu komendy GMBRWN. Można ją znaleźć tylko po zresetowaniu kursora za pomocą komendy GMBRWF (lub ponownego otwarcia kolejki).

Opcja GMMUC może być później używana z nieprzeglądaniem wywołania MQGET, jeśli jest to wymagane, w celu usunięcia komunikatu z kolejki.

Kursor przeglądania nie jest przenoszony przez nieprzeglądanie wywołań MQGET przy użyciu tego samego uchwytu *HOBJ*.

Opcja GMLK może zostać określona razem z tą opcją, aby spowodować zablokowanie komunikatu, który ma zostać zablokowany.

Parametr GMBRWN może być podany z dowolną poprawną kombinacją opcji GM* i MO*, które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli określono wartość GMLOGO, komunikaty są przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty są przeglądane w porządku fizycznym. Jeśli określono parametr GMBRWF, można przetaczać się między porządkiem logicznym a porządkiem fizycznym, ale kolejne wywołania MQGET używające komendy GMBRWN muszą przeglądać kolejkę w tej samej kolejności, co ostatnie wywołanie, które określił GMBRWF dla uchwytu kolejki. Wywołanie nie powiodło się z kodem przyczyny RC2259, jeśli ten warunek nie jest spełniony.

Uwaga: Jeśli wywołanie MQGET jest używane do przeglądania poza końcem grupy komunikatów (lub komunikatu logicznego nie w grupie), gdy nie określono programu GMLOGO, należy zachować szczególną ostrożność podczas wywołania MQGET. Jeśli na przykład ostatni komunikat w grupie jest poprzedzony pierwszym komunikatem w grupie w kolejce, użycie komendy GMBRWN do przejścia poza koniec grupy, podanie parametru MOSEQN z parametrem MDSEQ ustawionym na wartość 1 (w celu znalezienia pierwszego komunikatu następnej grupy) zwróci ponownie pierwszy komunikat w grupie, który został już przeglądany. Może to nastąpić natychmiast lub kilka wywołań MQGET w późniejszym czasie (jeśli istnieją grupy, które się do tego zdarzają).

Możliwość nieskończonej pętli można uniknąć, otwierając kolejkę dwukrotnie w celu przeglądania:

- Użyj pierwszego uchwytu, aby przeglądać tylko pierwszy komunikat w każdej grupie.
- Użyj drugiego uchwytu, aby przeglądać tylko komunikaty należące do określonej grupy.
- Użyj opcji MO*, aby przenieść drugi kursor przeglądania na pozycję pierwszego kursora przeglądania, przed przeglądaniem wiadomości w grupie.
- Nie należy używać komendy GMBRWN do przeglądania poza końcem grupy.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołań MQGET, które usuwają komunikaty z kolejki.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF
- GMBRWC
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMBRWC

Przeglądaj kursor pod kursorem przeglądania.

Ta opcja powoduje, że komunikat wskazuje, że kursor przeglądania zostanie pobrany bez zniszczenia, niezależnie od opcji MO* określonych w polu GMMO w produkcie MQGMO.

Komunikat wskazywał na kursor przeglądania, który był ostatnio wczytywał za pomocą opcji GMBRWF lub GMBRWN. Wywołanie nie powiedzie się, jeśli żadna z tych wywołań nie została wydana dla tej kolejki od momentu jej otwarcia lub jeśli komunikat, który był pod kursorem przeglądania, został pobrany w sposób destruktywny.

Pozycja kursora przeglądania nie jest zmieniana przez to wywołanie.

Opcja GMMUC może być następnie używana z nieprzeglądaniem wywołania MQGET, jeśli jest to wymagane, w celu usunięcia komunikatu z kolejki.

Kursor przeglądania nie jest przenoszony za pomocą wywołania MQGET bez przeglądania przy użyciu tego samego uchwytu *HOB*. Nie jest też przenoszone przez przeglądanie wywołania MQGET, które zwraca kod zakończenia CCFAIL, lub kod przyczyny RC2080.

Jeśli parametr GMBRWC jest określony z GMLK:

- Jeśli jest już zablokowany komunikat, musi to być ten, który znajduje się pod kursorem, tak aby został zwrócony bez odblokowania i ponownego zablokowania. Komunikat pozostaje zablokowany.
- Jeśli nie ma zablokowanego komunikatu, komunikat pod kursorem przeglądania (jeśli taki istnieje) jest zablokowany i zwrócony do aplikacji. Jeśli pod kursorem przeglądania nie ma komunikatu, wywołanie nie powiedzie się.

Jeśli parametr GMBRWC jest określony bez parametru GMLK:

- Jeśli jest już zablokowany komunikat, musi to być ten, który znajduje się pod kursorem. Ten komunikat jest zwracany do aplikacji, a następnie odblokowany. Ponieważ komunikat jest teraz odblokowany, nie ma gwarancji, że można go ponownie przeglądać lub pobrać destruktywnie (może być odtwarzany destruktywnie przez inną aplikację pobierającą komunikaty z kolejki).
- Jeśli nie ma zablokowanego komunikatu, do aplikacji zwracany jest komunikat pod kursorem przeglądania (jeśli taki istnieje). Jeśli pod kursorem przeglądania nie ma komunikatu, wywołanie nie powiedzie się.

Jeśli parametr GMCMPM jest określony z wartością GMBRWC, kursor przeglądania musi zidentyfikować komunikat z polem *MDOFF* w strukturze MQMD, które wynosi zero. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2246.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołań MQGET, które usuwają komunikaty z kolejki.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMMUC

Pobierz komunikat pod kursorem przeglądania.

Ta opcja powoduje, że komunikat wskazuje kursor przeglądania, który ma zostać pobrany, niezależnie od opcji MO* określonych w polu *GMMO* w produkcie MQGMO. Komunikat jest usuwany z kolejki.

Komunikat wskazywał na kursor przeglądania, który był ostatnio wczytywał za pomocą opcji GMBRWF lub GMBRWN.

Jeśli wartość GMCMPM jest określona za pomocą GMMUC, kursor przeglądania musi zidentyfikować komunikat z polem *MDOFF* w strukturze MQMD, które wynosi zero. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2246.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF
- GMBRWC
- GMBRWN
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta zarówno dla przeglądania, jak i dla danych wejściowych. Jeśli kursor przeglądania nie wskazuje aktualnie odtwarzalnego komunikatu, wywołanie MQGET zwróci błąd.

Opcje blokady: Następujące opcje odnoszą się do blokowania komunikatów w kolejce:

GMLK

Zablokuj wiadomość.

Ta opcja powoduje zablokowanie komunikatu, który jest przeglądany, dzięki czemu komunikat staje się niewidoczny dla wszystkich innych uchwytów otwartych dla kolejki. Tę opcję można określić tylko wtedy, gdy zostanie podana jedna z następujących opcji:

- GMBRWF
- GMBRWN
- GMBRWC

Dla każdego uchwytu kolejki może być zablokowany tylko jeden komunikat, ale może to być komunikat logiczny lub komunikat fizyczny:

- Jeśli określono parametr GMCMPM, wszystkie segmenty komunikatów, które tworzą komunikat logiczny, są zablokowane do uchwytu kolejki (jeśli są one wszystkie obecne w kolejce i dostępne do pobrania).
- Jeśli wartość GMCMPM nie jest określona, tylko jeden komunikat fizyczny jest zablokowany dla uchwytu kolejki. Jeśli ten komunikat ma być segmentem komunikatu logicznego, zablokowany segment uniemożliwia innym aplikacjom użycie GMCMPM w celu pobrania lub przeglądania komunikatu logicznego.

Zablokowany komunikat jest zawsze wyświetlany pod kursorem przeglądania, a komunikat może zostać usunięty z kolejki za pomocą późniejszego wywołania MQGET, które określa opcję GMMUC. Inne wywołania MQGET używające uchwytu kolejki również mogą usunąć komunikat (na przykład wywołanie, które określa identyfikator komunikatu zablokowanego komunikatu).

Jeśli wywołanie zwróci kod zakończenia CCFAIL lub CCWARN z kodem przyczyny RC2080, żaden komunikat nie jest zablokowany.

Jeśli aplikacja zdecyduje, że komunikat nie zostanie usunięty z kolejki, blokada jest zwolniona przez:

- Wywołanie innego wywołania MQGET dla tego uchwytu, z określonym GMBRWF lub GMBRWN (z lub bez GMLK); komunikat jest odblokowany, jeśli wywołanie zakończy się z CCOK lub CCWARN, ale pozostaje zablokowane, jeśli wywołanie zakończy się z CCFAIL. Istnieją jednak wyjątki:
 - Komunikat nie jest odblokowany, jeśli funkcja CCWARN jest zwracana z kodem RC2080.
 - Komunikat zostanie odblokowany, jeśli kod CCFAIL zostanie zwrócony z kodem RC2033.

Jeśli określono również wartość GMLK, zwrócony komunikat jest zablokowany. Jeśli wartość GMLK nie zostanie określona, po wywołaniu nie zostanie wyświetlony żaden zablokowany komunikat.

Jeśli określono GMWT, a żaden komunikat nie jest dostępny natychmiast, odblokowanie na oryginalnym komunikacie następuje przed rozpoczęciem oczekiwania (pod warunkiem, że wywołanie jest wolne od błędów).

- Wywołanie innego wywołania MQGET dla tego uchwytu z produktem GMBRWC (bez GMLK); komunikat jest odblokowany, jeśli wywołanie zakończy się z CCOK lub CCWARN, ale pozostaje zablokowane, jeśli wywołanie zakończy się z CCFAIL. Jednak zastosowanie ma następujący wyjątek:

- Komunikat nie jest odblokowany, jeśli funkcja CCWARN jest zwracana z kodem RC2080.
- Wywołanie innego wywołania MQGET dla tego uchwytu z GMUNLK.
- Wywołanie wywołania MQCLOSE dla tego uchwytu (jawnego lub niejawnie przez zakończenie aplikacji).

Aby określić tę opcję, inną niż OOBROW, która jest potrzebna w celu określenia towarzyszącej opcji przeglądania, nie jest wymagana żadna specjalna opcja otwarcia.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMSYP
- GMPSYP
- GMUNLK

GMUNLK

Odblokuj wiadomość.

Komunikat, który ma zostać odblokowany, musi być wcześniej zablokowany przez wywołanie MQGET z opcją GMLK. Jeśli dla tego uchwytu nie jest zablokowany żaden komunikat, wywołanie kończy się z CCWARN i RC2209.

Parametry *MSGDSC*, *BUFLN*, *BUFFERi* *DATLEN* nie są sprawdzane ani zmieniane, jeśli określono parametr GMUNLK. W programie *BUFFER* nie jest zwracany żaden komunikat.

Do określenia tej opcji nie jest wymagana żadna specjalna opcja otwarcia (choć OOBROW jest potrzebne do wystawienia żądania blokady w pierwszej kolejności).

Ta opcja nie jest poprawna z następującymi opcjami, z wyjątkiem następujących:

- GMNWT
- GMNSYP

Obie te opcje są przyjmowane, niezależnie od tego, czy zostały określone, czy nie.

Opcje danych komunikatu: Następujące opcje odnoszą się do przetwarzania danych komunikatu, gdy komunikat jest odczytywany z kolejki:

GMATM

Zezwalaj na obcinanie danych komunikatu.

Jeśli bufor komunikatów jest zbyt mały, aby pomieścić pełny komunikat, ta opcja umożliwia wywołanie MQGET w celu zapewnienia buforu jako dużą część komunikatu, ponieważ bufor może być wstrzymany, należy wprowadzić kod zakończenia ostrzeżenia i zakończyć przetwarzanie. Oznacza to:

- Podczas przeglądania komunikatów kursor przeglądania jest zaawansowany do zwróconego komunikatu.
- Podczas usuwania komunikatów zwrócony komunikat jest usuwany z kolejki.
- Kod przyczyny RC2079 jest zwracany, jeśli nie wystąpi inny błąd.

Bez tej opcji bufor jest nadal wypełniany tak dużą ilością komunikatu, jaki może być wstrzymany, generowany jest kod zakończenia ostrzeżenia, ale przetwarzanie nie jest zakończone. Oznacza to:

- Podczas przeglądania komunikatów kursor przeglądania nie jest zaawansowany.
- Podczas usuwania komunikatów komunikat nie jest usuwany z kolejki.
- Kod przyczyny RC2080 jest zwracany, jeśli nie wystąpi inny błąd.

GMCONV

Konwersja danych komunikatu.

Ta opcja żąda, aby dane aplikacji w komunikacie były przekształcane, aby były zgodne z wartościami *MDCSI* i *MDENC* określonymi w parametrze *MSGDSC* w wywołaniu MQGET, zanim dane zostaną skopiowane do parametru *BUFFER*.

Pole *MDFMT* określone podczas umieszczania komunikatu jest przejęte przez proces konwersji w celu określenia rodzaju danych w komunikacie. Konwersja danych komunikatu jest przez menedżer kolejek dla formatów wbudowanych, a przez użytkownika-wyjście zapisane dla innych formatów.

- Jeśli konwersja zostanie wykonana pomyślnie, pola *MDCSI* i *MDENC* określone w parametrze *MSGDSC* nie zostaną zmienione podczas powrotu z wywołania *MQGET*.
- Jeśli konwersja nie może zostać wykonana pomyślnie (ale wywołanie *MQGET* w inny sposób zakończy się bez błędu), dane komunikatu są zwracane bez konwersji, a pola *MDCSI* i *MDENC* w programie *MSGDSC* są ustawione na wartości dla komunikatu bez konwersji. W tym przypadku kodem zakończenia jest *CCWARN*.

W obu przypadkach w tych polach opisano identyfikator zestawu znaków i kodowanie danych komunikatu zwracanych w parametrze *BUFFER*.

See the *MDFMT* field described in [“MQMD-deskryptor komunikatu”](#) na stronie 1124 for a list of format names for which the queue manager performs the conversion.

Opcje grupy i segmentu: Następujące opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Te definicje mogą być pomocne w zrozumieniu opcji:

Komunikat fizyczny

Jest to najmniejsza jednostka informacji, która może zostać umieszczona w kolejce lub usunięta z kolejki. Jest ona często zgodna z informacjami podanymi lub pobraną w pojedynczej operacji *MQPUT*, *MQPUT1* lub *MQGET*. Każdy komunikat fizyczny ma własny deskryptor komunikatu (*MQMD*). Ogólnie, komunikaty fizyczne wyróżniają się różnymi wartościami dla identyfikatora komunikatu (pole *MDMID* w strukturze *MQMD*), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny

Jest to pojedyncza jednostka informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny byłby taki sam, jak komunikat fizyczny. Jeśli jednak komunikaty logiczne są duże, ograniczenia systemowe mogą spowodować konieczność lub konieczność podzielenia komunikatu logicznego na dwa lub więcej komunikatów fizycznych nazywanych segmentami.

Komunikat logiczny, który został posegmentowany, składa się z dwóch lub większej liczby komunikatów fizycznych o tym samym identyfikatorze grupy niepustych (pole *MDGID* w strukturze *MQMD*) i o tym samym numerze kolejnym komunikatu (pole *MDSEQ* w strukturze *MQMD*). Segmenty są rozróżniane przez różne wartości dla przesunięcia segmentu (pole *MDOFF* w strukturze *MQMD*), co daje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dopuszczona przez aplikację wysyłającą, ma również identyfikator grupy niezerowej, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają identyfikator grupy o wartości *NULL* (*GINONE*), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy

Jest to zestaw jednego lub większej liczby komunikatów logicznych, które mają ten sam niepusty identyfikator grupy. Komunikaty logiczne w grupie są rozróżniane przez różne wartości dla numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do *n*, gdzie *n* jest liczbą komunikatów logicznych w grupie. Jeśli co najmniej jeden komunikat logiczny jest segmentowany, w grupie jest więcej niż *n* komunikatów fizycznych.

GMLOGO

Komunikaty w grupach i segmentach komunikatów logicznych są zwracane w porządku logicznym.

Ta opcja służy do określania kolejności, w jakiej komunikaty są zwracane przez kolejne wywołania *MQGET* dla uchwytu kolejki. Opcja musi być określona w każdym z tych wywołań, aby mieć efekt.

Jeśli określono parametr GMLOGO dla kolejnych wywołań MQGET dla uchwytu kolejki, komunikaty w grupach są zwracane w kolejności podanej przez ich numery kolejne komunikatów, a segmenty komunikatów logicznych są zwracane w kolejności określonej przez ich przesunięcia segmentów. Kolejność ta może różnić się od kolejności, w jakiej komunikaty i segmenty występują w kolejce.

Uwaga: Określenie GMLOGO nie ma niekorzystnych skutków dla komunikatów, które nie należą do grup, a które nie są segmentami. W efekcie takie komunikaty są traktowane tak, jakby każdy należał do grupy komunikatów składającej się tylko z jednego komunikatu. W ten sposób można bezpiecznie określić GMLOGO podczas pobierania komunikatów z kolejek, które mogą zawierać kombinację komunikatów w grupach, segmentach komunikatów i nieposegmentowanych komunikatów w grupach.

Aby zwrócić komunikaty w wymaganej kolejności, menedżer kolejek zachowuje informacje o grupach i segmentach między kolejnymi wywołaniami MQGET. Informacje te identyfikują bieżącą grupę komunikatów i bieżący komunikat logiczny dla uchwytu kolejki, bieżącą pozycję w grupie i komunikat logiczny oraz informację, czy komunikaty są pobierane w ramach jednostki pracy. Ponieważ menedżer kolejek zachowuje te informacje, aplikacja nie musi ustawiać informacji o grupach i segmentach przed każdym wywołaniem MQGET. W szczególności oznacza to, że aplikacja nie musi ustawiać pól *MDGID*, *MDSEQi* *MDOFF* w strukturze MQMD. Jednak aplikacja musi poprawnie ustawić opcję GMSYP lub GMNSYP dla każdego wywołania.

Po otwarciu kolejki nie ma bieżącej grupy komunikatów i nie ma bieżącego komunikatu logicznego. Grupa komunikatów staje się bieżącą grupą komunikatów, gdy komunikat z flagą MFMIIG jest zwracany przez wywołanie MQGET. Jeśli GMLOGO jest określone w kolejnych wywołaniach, ta grupa pozostaje grupą bieżącą do momentu zwrócenia komunikatu o następującej treści:

- MFLMIG bez MFSEG (czyli ostatni komunikat logiczny w grupie nie jest segmentowany), lub
- MFLMIG z MFLSEG (to znaczy, że zwrócony komunikat jest ostatnim segmentem ostatniego komunikatu logicznego w grupie).

Gdy taki komunikat zostanie zwrócony, grupa komunikatów zostanie zakończona, a po pomyślnym zakończeniu wywołania MQGET nie istnieje już bieżąca grupa. W podobny sposób komunikat logiczny staje się bieżącym komunikatem logicznym, gdy komunikat z flagą MFSEG jest zwracany przez wywołanie MQGET, a komunikat logiczny zostaje zakończony, gdy zostanie zwrócony komunikat z flagą MFLSEG.

Jeśli nie zostaną określone żadne kryteria wyboru, kolejne wywołania MQGET zwracają (w poprawnej kolejności) komunikaty dla pierwszej grupy komunikatów w kolejce, a następnie komunikaty dla drugiej grupy komunikatów itd. aż do momentu, w którym nie będzie dostępnych więcej komunikatów. Możliwe jest wybranie wybranych grup komunikatów, określając jedną lub więcej spośród następujących opcji w polu *GMMO* :

- MOMSGI
- MOCORI
- MOGRPI

Opcje te są jednak skuteczne tylko wtedy, gdy nie ma bieżącej grupy komunikatów lub komunikatu logicznego. Patrz pole *GMMO* opisane w tym temacie.

W programie Tabela 143 na stronie 1103 wyświetlane są wartości pól *MDMID*, *MDCID*, *MDGID*, *MDSEQi* *MDOFF* , dla których menedżer kolejek szuka podczas próby znalezienia komunikatu, który ma zostać zwrócony w wywołaniu MQGET. Ma to zastosowanie zarówno do usuwania komunikatów z kolejki, jak i do przeglądania komunikatów w kolejce. Kolumny w tabeli mają następujące znaczenia:

PROTOKÓŁ ORD

Wskazuje, czy opcja GMLOGO jest określona w wywołaniu.

Grp

Wskazuje, czy bieżąca grupa komunikatów istnieje przed wywołaniem.

Komunikat dziennika Cur

Wskazuje, czy bieżący komunikat logiczny istnieje przed wywołaniem.

Pozostałe kolumny

Pokaż wartości, dla których szuka się menedżer kolejek. "Wstecz" oznacza wartość zwracaną dla pola w poprzednim komunikacie dla uchwytu kolejki.

Tabela 143. Opcje MQGET odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych							
Opcje określone przez użytkownika	Status grupy i dziennika-komunikat przed wywołaniem		Wartości, dla których menedżer kolejek szuka				
PROTO KÓŁ ORD	Cur grp	Komunikat dziennika Cur	MDMID	MDCID	MDGID	MDSEQ	MDOFF
Tak	Nie	Nie	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO	1	0
Tak	Nie	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + długość poprzedniego segmentu
Tak	Tak	Nie	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + długość poprzedniego segmentu
Nie	Albo	Albo	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO	Sterowane przez produkt GMMO

Jeśli w kolejce znajduje się wiele grup komunikatów, które mogą zostać zwrócone, grupy są zwracane w kolejności określonej przez pozycję w kolejce pierwszego segmentu pierwszego komunikatu logicznego w każdej grupie (czyli komunikaty fizyczne, które mają numery kolejne komunikatów 1, oraz przesunięcia 0, określają kolejność, w jakiej zwracane są grupy).

Opcja GMLOGO wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy komunikat logiczny lub segment w grupie jest pobierany w jednostce pracy, wszystkie pozostałe komunikaty i segmenty w grupie muszą zostać pobrane w ramach jednostki pracy, o ile ten sam uchwyt kolejki jest używany. Nie muszą one jednak być pobierane w ramach tej samej jednostki pracy. Umożliwia to grupie komunikatów składającej się z wielu komunikatów fizycznych, które mają zostać podzielone na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy komunikat logiczny lub segment w grupie *nie* jest pobierany w jednostce pracy, żaden z pozostałych komunikatów logicznych i segmentów w grupie nie może być pobrany w jednostce pracy, jeśli ten sam uchwyt kolejki jest używany.

Jeśli te warunki nie zostaną spełnione, wywołanie MQGET nie powiedzie się z kodem przyczyny RC2245.

Jeśli określono parametr GMLOGO, wartość MQGMO podana w wywołaniu MQGET nie może być mniejsza niż wartość GMVER2, a wartość MQMD nie może być mniejsza niż MDVER2. Jeśli ten warunek nie jest spełniony, wywołanie nie powiedzie się z kodem przyczyny RC2256 lub RC2257, jeśli jest to właściwe.

Jeśli GMLOGO nie jest określone dla kolejnych wywołań MQGET dla uchwytu kolejki, zwracane są komunikaty bez względu na to, czy należą one do grup komunikatów, czy też są segmentami komunikatów logicznych. Oznacza to, że komunikaty lub segmenty z określonej grupy lub komunikatu logicznego mogą zostać zwrócone poza kolejką lub mogą być połączone z komunikatami lub segmentami z innych grup lub z komunikatów logicznych albo z komunikatami, które nie znajdują się w grupach i nie są segmentami. W takiej sytuacji konkretne komunikaty zwracane przez kolejne wywołania MQGET są kontrolowane przez opcję MO* określone w tych wywołaniach (szczegółowe informacje na temat tych opcji zawiera opis pola GMMO (opisany w sekcji [“MQGMO-Opcje Get-message”](#) na stronie 1089).

Jest to technika, której można użyć do zrestartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić pola MDGID, MDSEQ, MDOFFi GMMO na odpowiednie wartości, a następnie wywołać wywołanie MQGET z zestawem GMSYP lub GMNSYP w razie potrzeby, ale bez określania parametru GMLOGO. Jeśli to wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupie i segmencie, a kolejne wywołania MQGET używające tego uchwytu kolejki mogą określać wartość GMLOGO jako normalną.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołania MQGET, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołania MQPUT. Dodatkowo menedżer kolejek zachowuje oddzielne informacje dla:

- Wywołania MQGET, które usuwają komunikaty z kolejki.
- Wywołania MQGET, które przeglądają komunikaty w kolejce.

W przypadku dowolnego uchwytu kolejki aplikacja może łączyć wywołania MQGET, które określają GMLOGO z wywołaniami MQGET, które nie są, ale należy zauważyć następujące punkty:

- Jeśli wartość GMLOGO nie zostanie określona, każde pomyślne wywołanie MQGET powoduje, że menedżer kolejek ustawia zapisane informacje o grupach i segmentach na wartości odpowiadające zwróconej wiadomości; zastępuje to istniejące informacje o grupach i segmentach zachowane przez menedżer kolejek dla uchwytu kolejki. Modyfikowane są tylko informacje odpowiednie do działania wywołania (przeglądanie lub usuwanie).
- Jeśli wartość GMLOGO nie zostanie określona, wywołanie nie powiedzie się, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może jednak zakończyć się powodzeniem z kodem zakończenia CCWARN. Tabela 144 na stronie 1104 przedstawia różne przypadki, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia nie jest kodem CCOK, kodem przyczyny jest jeden z następujących kodów:
 - RC2241
 - RC2242
 - RC2245

Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie podczas przeglądania kolejki lub podczas zamykania kolejki, która została otwarta do przeglądania, ale nie jest wprowadzana; w takich przypadkach kod zakończenia jest zawsze CCOK (zakładając, że nie występują inne błędy).

Tabela 144. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach

Bieżące połączenie to	Poprzednie wywołanie to MQGET z GMLOGO	Poprzednie wywołanie to MQGET bez GMLOGO
MQGET z GMLOGO	CCFAIL	CCFAIL

Tabela 144. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach (kontynuacja)

Bieżące połączenie to	Poprzednie wywołanie to MQGET z GMLOGO	Poprzednie wywołanie to MQGET bez GMLOGO
MQGET bez GMLOGO	CCWARN	CCOK
MQCLOSE z niezakończonym komunikatem grupowym lub logicznym	CCWARN	CCOK

Aplikacje, które po prostu chcą pobierać komunikaty i segmenty w porządku logicznym, są zalecane w celu określenia GMLOGO, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia aplikację z potrzeby zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tą informacją. Jednak w przypadku wyspecjalizowanych aplikacji może być potrzebna większa kontrola niż podana w opcji GMLOGO, co może zostać osiągnięte przez niepodanie tej opcji. W takim przypadku aplikacja musi upewnić się, że pola *MDMID*, *MDCID*, *MDGID*, *MDSEQ* i *MDOFF* w strukturze *MQMD* oraz opcje *MO** w produkcie *GMMO* w produkcie *MQGMO* są ustawione poprawnie, przed każdym wywołaniem *MQGET*.

Na przykład aplikacja, która chce przekazywać wiadomości fizyczne, które otrzymuje, bez względu na to, czy te komunikaty znajdują się w grupach, czy w segmentach komunikatów logicznych, nie powinna określać GMLOGO. Jest to spowodowane tym, że w złożonej sieci z wieloma ścieżkami między wysyłającym i odbierającym menedżery kolejek, komunikaty fizyczne mogą być odbierane poza kolejnością. Jeśli nie określono parametru GMLOGO i odpowiedniego narzędzia *PMLOGO* w wywołaniu *MQPUT*, aplikacja przekazujący może pobrać i przesłać każdy komunikat fizyczny zaraz po jego nadejściu, bez konieczności oczekiwania na nadejście kolejnego komunikatu w kolejności logicznej.

GMLOGO można określić przy użyciu dowolnej z pozostałych opcji *GM**, a w odpowiednich okolicznościach z różnymi opcjami *MO**.

GMCMPM

Wczytywane są tylko kompletne komunikaty logiczne.

Ta opcja określa, że wywołanie *MQGET* może zwrócić tylko pełny komunikat logiczny. Jeśli komunikat logiczny jest segmentowany, menedżer kolejek ponownie zestawia segmenty i zwraca do aplikacji pełny komunikat logiczny. Fakt, że komunikat logiczny był podzielony na segmenty, nie jest widoczny dla aplikacji pobierających dane.

Uwaga: Jest to jedyna opcja, która powoduje, że menedżer kolejek ma ponownie składać segmenty komunikatów. Jeśli ta opcja nie zostanie określona, segmenty są zwracane pojedynczo do aplikacji, jeśli są obecne w kolejce (i spełniają inne kryteria wyboru określone w wywołaniu *MQGET*). Aplikacje, które nie chcą otrzymywać poszczególnych segmentów, powinny w związku z tym zawsze określać GMCMPM.

Aby użyć tej opcji, aplikacja musi udostępnić bufor, który jest wystarczająco duży, aby pomieścić pełny komunikat, lub określić opcję *GMATM*.

Jeśli kolejka zawiera segmentowane komunikaty z brakującą część segmentów (być może dlatego, że zostały one opóźnione w sieci i jeszcze nie dotarły), podanie parametru GMCMPM zapobiega pobieraniu segmentów należących do niekompletnych komunikatów logicznych. Jednak te segmenty komunikatów nadal przyczyniają się do wartości atrybutu kolejki produktu *CurrentQDepth*. Oznacza to, że nie mogą istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość *CurrentQDepth* jest większa od zera.

W przypadku trwałych komunikatów menedżer kolejek może ponownie złożyć segmenty tylko w ramach jednostki pracy:

- Jeśli wywołanie *MQGET* działa w ramach jednostki pracy zdefiniowanej przez użytkownika, ta jednostka pracy jest używana. Jeśli wywołanie nie powiedzie się w ramach procesu ponownego składania, menedżer kolejek przywraca w kolejce wszystkie segmenty usunięte

podczas ponownego składania. Jednak niepowodzenie nie uniemożliwia pomyślnego wykonania jednostki pracy.

- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, a nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie zakończy się pomyślnie, menedżer kolejek zatwierdza jednostkę pracy automatycznie (aplikacja nie musi wykonywać tej czynności). Jeśli wywołanie nie powiedzie się, menedżer kolejek wytworzy kopię zapasową jednostki pracy.
- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, ale nie istnieje zdefiniowana przez użytkownika jednostka pracy, menedżer kolejek nie może przeprowadzić ponownego składania. Jeśli komunikat nie wymaga ponownego składania, wywołanie może się jeszcze powieść. Jeśli jednak komunikat wymaga ponownego składania, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2255 .

W przypadku komunikatów nietrwałych menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu przeprowadzenia ponownego montażu.

Każdy komunikat fizyczny, który jest segmentem, ma własny deskryptor komunikatu. W przypadku segmentów tworzących pojedynczy komunikat logiczny większość pól w deskrytorze komunikatu jest taka sama dla wszystkich segmentów w komunikacie logicznym-zwykle jest to tylko pola *MDMID*, *MDOFF* i *MDMFL* , które różnią się między segmentami w komunikacie logicznym. Jeśli jednak segment jest umieszczany w kolejce niedostarczonych komunikatów w pośrednim menedżerze kolejek, procedura obsługi DLQ pobiera komunikat z opcją *GMCONV* i może to spowodować zmianę zestawu znaków lub kodowania tego segmentu. Jeśli procedura obsługi DLQ pomyślnie wyśle segment na swój sposób, segment może mieć zestaw znaków lub kodowanie, które różnią się od innych segmentów w komunikacie logicznym, gdy segment zostanie ostatecznie odebrany do docelowego menedżera kolejek.

Komunikat logiczny składający się z segmentów, w których *MDCSI*, *MDENCL* lub obu tych pól nie mogą być ponownie składane przez menedżera kolejek w pojedynczy komunikat logiczny. Zamiast tego menedżer kolejek jest ponownie montuje i zwraca pierwsze kilka kolejnych segmentów na początku komunikatu logicznego, które mają takie same identyfikatory i kodowania zestawu znaków, a wywołanie *MQGET* kończy się odpowiednio kodem zakończenia *CCWARN* i kodem przyczyny *RC2243* lub *RC2244* . Dzieje się tak niezależnie od tego, czy określono *GMCONV*. Aby pobrać pozostałe segmenty, aplikacja musi ponownie wywołać wywołanie *MQGET* bez opcji *GMCPMP*, pobierając segmenty jeden za pomocą jednego z nich. Program *GMLOGO* może być używany do pobierania pozostałych segmentów w kolejności.

Możliwe jest również, że aplikacja umieszcza segmenty w celu ustawienia innych pól w deskrytorze komunikatu na wartości, które różnią się między segmentami. Jednak nie ma żadnej korzyści, jeśli aplikacja odbierający używa *GMCPMP* do pobrania komunikatu logicznego. Gdy menedżer kolejek przedstawia komunikat logiczny, w deskrytorze komunikatu zwracane są wartości z deskryptora komunikatu dla pierwszego segmentu. Jedyny wyjątek to pole *MDMFL* , które wskazuje menedżer kolejek w celu wskazania, że zmontowany komunikat jest jedynym segmentem.

Jeśli określono parametr *GMCPMP* dla komunikatu raportu, menedżer kolejek wykonuje przetwarzanie specjalne. Menedżer kolejek sprawdza kolejkę w celu sprawdzenia, czy wszystkie komunikaty raportu tego typu odnoszące się do różnych segmentów w komunikacie logicznym znajdują się w kolejce. Jeśli są one dostępne, można je pobrać jako pojedynczy komunikat, określając *GMCPMP*. Aby to było możliwe, komunikaty raportu muszą być generowane przez menedżer kolejek lub agent *MCA* obsługujący segmentację albo aplikacja źródłowa musi zażądać co najmniej 100 bajtów danych komunikatu (to znaczy należy określić odpowiednie opcje *RO** *D* lub *RO** *F*). Jeśli dla segmentu jest mniejsza niż pełna ilość danych aplikacji, brakujące bajty są zastępowane wartościami pustymi w zwróconej komunikacie raportu.

Jeśli wartość *GMCPMP* jest określona za pomocą *GMMUC* lub *GMBRWC*, kursor przeglądania musi być umieszczony w komunikacie z polem *MDOFF* w strukturze *MQMD*, które ma wartość 0. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny *RC2246* .

GMCMPPM implikuje GMASGA, które nie są w związku z tym określone.

Program GMCMPPM może być określony z innymi opcjami środowiska GM* poza GMPSYP, a także z dowolnymi opcjami MO* poza MOOFFS.

GMAMSA

Wszystkie komunikaty w grupie muszą być dostępne.

Ta opcja określa, że komunikaty w grupie stają się dostępne do pobrania tylko wtedy, gdy wszystkie komunikaty w grupie są dostępne. Jeśli kolejka zawiera grupy komunikatów z niektórymi brakami komunikatów (być może dlatego, że zostały one opóźnione w sieci i jeszcze nie dotarły), określenie GMAMSA uniemożliwia pobranie komunikatów należących do niekompletnych grup. Jednak te komunikaty nadal przyczyniają się do wartości atrybutu kolejki produktu *CurrentQDepth*. Oznacza to, że nie mogą istnieć żadne możliwe do pobrania grupy komunikatów, nawet jeśli wartość *CurrentQDepth* jest większa od zera. Jeśli nie ma innych komunikatów, które są dostępne do pobrania, kod przyczyny RC2033 jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie GMAMSA zależy od tego, czy GMLOGO jest również określone:

- Jeśli określono obie opcje, GMAMSA ma wpływ na *tylko*, gdy nie ma żadnej bieżącej grupy lub komunikatu logicznego. Jeśli istnieje *jest* bieżąca grupa lub komunikat logiczny, GMAMSA jest ignorowana. Oznacza to, że GMAMSA może pozostawać w trakcie przetwarzania komunikatów w kolejności logicznej.
- Jeśli wartość GMAMSA jest określona bez GMLOGO, GMAMSA zawsze ma działanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego komunikatu w grupie, aby można było usunąć pozostałe komunikaty w grupie.

Pomyślne zakończenie wywołania MQGET z określeniem GMAMSA oznacza, że w momencie wywołania wywołania MQGET wszystkie komunikaty w grupie znajdowały się w kolejce. Należy jednak pamiętać, że inne aplikacje nadal są w stanie usunąć komunikaty z grupy (grupa nie jest zablokowana dla aplikacji, która pobiera pierwszy komunikat w grupie).

Jeśli ta opcja nie zostanie podana, komunikaty należące do grup będą mogły być pobierane nawet wtedy, gdy grupa jest niekompletna.

GMAMSA implikuje GMASGA, które nie muszą być określone.

GMAMSA może być określony z dowolną z pozostałych opcji środowiska GM*, a także z dowolną z opcji MO*.

GMASGA

Wszystkie segmenty w komunikacie logicznym muszą być dostępne.

Ta opcja określa, że segmenty w komunikacie logicznym stają się dostępne do pobrania tylko wtedy, gdy wszystkie segmenty w komunikacie logicznym są dostępne. Jeśli kolejka zawiera segmentowane komunikaty z brakującą część segmentów (być może dlatego, że zostały one opóźnione w sieci i jeszcze nie dotarły), określenie GMASGA uniemożliwia pobieranie segmentów należących do niekompletnych komunikatów logicznych. Jednak segmenty te nadal przyczyniają się do wartości atrybutu kolejki produktu *CurrentQDepth*. Oznacza to, że nie można pobrać komunikatów logicznych, nawet jeśli wartość *CurrentQDepth* jest większa od zera. Jeśli nie ma innych komunikatów, które są dostępne do pobrania, kod przyczyny RC2033 jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie GMASGA zależy od tego, czy GMLOGO jest również określone:

- Jeśli określono obie opcje, GMASGA ma działanie tylko wtedy, gdy nie ma bieżącego komunikatu logicznego. Jeśli istnieje bieżący komunikat logiczny, wartość GMASGA jest ignorowana. Oznacza to, że GMASGA może pozostać w trakcie przetwarzania komunikatów w kolejności logicznej.
- Jeśli parametr GMASGA zostanie określony bez GMLOGO, GMASGA zawsze ma działanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego segmentu

w komunikacie logicznym, aby możliwe było usunięcie pozostałych segmentów w komunikacie logicznym.

Jeśli ta opcja nie zostanie podana, segmenty komunikatów mogą być pobierane nawet wtedy, gdy komunikat logiczny jest niekompletny.

Podczas gdy zarówno GMCMPM, jak i GMASGA wymagają, aby wszystkie segmenty były dostępne, zanim można będzie pobrać dowolny z nich, to pierwsza z nich zwraca cały komunikat, podczas gdy ten ostatni umożliwia pobranie segmentów po jednym.

Jeśli dla komunikatu raportu określono GMASGA, menedżer kolejek wykonuje przetwarzanie specjalne. Menedżer kolejek sprawdza kolejkę w celu sprawdzenia, czy istnieje co najmniej jeden komunikat raportu dla każdego z segmentów, które składają się na pełny komunikat logiczny. Jeśli istnieje, warunek GMASGA jest spełniony. Jednak menedżer kolejek nie sprawdza typu komunikatów raportu, dlatego może istnieć mieszanka typów raportów w komunikatach raporty/dotyczących segmentów komunikatu logicznego. W wyniku tego powodzenie GMASGA nie oznacza, że GMCMPM zakończy się powodzeniem. Jeśli istnieje mieszanka typów raportów obecnych dla segmentów konkretnego komunikatu logicznego, te komunikaty muszą zostać pobrane jeden po jednym.

GMASGA może być określony z dowolną inną opcją GM*, a z dowolną z opcji MO*.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

GMNONE

Nie określono żadnych opcji.

Ta wartość może być używana do wskazania, że nie zostały określone żadne inne opcje. Wszystkie opcje przyjmują wartości domyślne. Parametr GMNONE jest zdefiniowany w dokumentacji programu pomocowego; nie jest przeznaczony, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

Wartością początkową pola *GMOPT* jest GMNWT.

GMRE1 (1 bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż GMVER2.

GMRL (10-cyfrowa liczba całkowita ze znakiem)

Długość zwracanych danych komunikatu (w bajtach).

Jest to pole wyjściowe, które jest ustawiane przez menedżer kolejek na długość w bajtach danych komunikatu zwróconych przez wywołanie MQGET w parametrze *BUFFER*. Jeśli menedżer kolejek nie obsługuje tej możliwości, parametr *GMRL* jest ustawiony na wartość RLUNDF.

Gdy komunikaty są przekształcane między kodowaniami lub zestawami znaków, dane komunikatu mogą czasami zmieniać wielkość. Po powrocie z wywołania MQGET:

- Jeśli parametr *GMRL* nie jest ustawiony na RLUNDF, liczba bajtów zwracanych przez dane komunikatu jest podawana przez produkt *GMRL*.
- Jeśli parametr *GMRL* ma wartość RLUNDF, liczba bajtów zwracanych przez dane komunikatu jest zwykle mniejsza od wartości *BUFLN* i *DATLEN*, ale może być mniejsza od tej, jeśli wywołanie MQGET zakończy się z kodem przyczyny RC2079. Jeśli tak się stanie, nieistotne bajty w parametrze *BUFFER* są ustawione na wartości NULL.

Zdefiniowane są następujące wartości specjalne:

RLUNDF

Długość zwróconych danych nie została zdefiniowana.

Wartością początkową tego pola jest RLUNDF. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż GMVER3.

GMRQN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa kolejki docelowej.

Jest to pole wyjściowe, które jest ustawiane przez menedżer kolejek na nazwę lokalną kolejki, z której został pobrany komunikat, zgodnie z definicją menedżera kolejek lokalnych. Wartość ta różni się od nazwy używanej do otwarcia kolejki, jeśli:

- Kolejka aliasowa została otwarta (w takim przypadku nazwa kolejki lokalnej, do której został zwrócony alias, jest zwracana), lub
- Otwarto kolejkę modelową (w takim przypadku zwracana jest nazwa dynamicznej kolejki lokalnej).

Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

GMR52 (1 bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż *GMVER4*.

GMSEG (1 bajtowy łańcuch znaków)

Flaga wskazująca, czy dozwolona jest dalsza segmentacja dla pobranego komunikatu.

Ma jedną z następujących wartości:

SEGIHB

Segmentacja nie jest dozwolona.

SEGALW

Segmentacja jest dozwolona.

To jest pole wyjściowe. Wartością początkową tego pola jest *SEGIHB*. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż *GMVER2*.

MSG1 (10-cyfrowa liczba całkowita ze znakiem)

Sygnał.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

MSG2 (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator sygnału.

Jest to pole zastrzeżone; jego wartość nie jest znacząca.

GMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

GMSIDV

Identyfikator struktury opcji get-message.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest *GMSIDV*.

GMSST (1 bajtowy łańcuch znaków)

Flaga wskazująca, czy pobrano komunikat, czy jest to segment komunikatu logicznego.

Ma jedną z następujących wartości:

SSNSEG

Komunikat nie jest segmentem.

SSSEG

Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

SSLSEG

Komunikat jest ostatnim segmentem komunikatu logicznego.

Jest to także wartość zwracana, jeśli komunikat logiczny składa się tylko z jednego segmentu.

To pole jest polem wyjściowym. Wartością początkową tego pola jest SSNSEG. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż GMVER2.

GMTOK (16-bajtowy łańcuch bitowy)

Token komunikatu.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Zdefiniowane są następujące wartości specjalne:

MTKNON

Brak znacznika komunikatu.

Wartość jest binarna zero dla długości pola.

Długość tego pola jest podana przez LNMTOK. Wartością początkową tego pola jest MTKNON. To pole jest ignorowane, jeśli parametr *GMVER* jest mniejszy niż GMVER3.

GMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

GMVER1

Struktura opcji komunikatu get-message w wersji Version-1 .

GMVER2

Struktura opcji get-message w wersji Version-2 .

GMVER3

Struktura opcji komunikatu get-message w wersji Version-3 .

GMVER4

Struktura opcji komunikatu get-message w wersji Version-4 .

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

GMVERC

Bieżąca wersja struktury opcji get-message.

To pole jest zawsze polem wejściowym. Początkowa wartość tego pola to GMVER1.

GMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

GMVER1

Struktura opcji komunikatu get-message w wersji Version-1 .

GMVER2

Struktura opcji get-message w wersji Version-2 .

GMVER3

Struktura opcji komunikatu get-message w wersji Version-3 .

GMVER4

Struktura opcji komunikatu get-message w wersji Version-4 .

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

GMVERC

Bieżąca wersja struktury opcji get-message.

To pole jest zawsze polem wejściowym. Początkowa wartość tego pola to GMVER1.

GMWI (10-cyfrowa liczba całkowita ze znakiem)

Interwał oczekiwania.

Jest to przybliżony czas wyrażony w milisekundach, przez który wywołanie MQGET oczekuje na nadejście odpowiedniego komunikatu (to znaczy komunikat spełniający kryteria wyboru określone w parametrze MSGDSC wywołania MQGET; więcej informacji zawiera opis pola MDMID opisany w sekcji "MQMD-deskryptor komunikatu" na stronie 1124). Jeśli po upływie tego czasu nie pojawił się odpowiedni komunikat, wywołanie zakończy się z kodem powrotu CCFAIL i kodem przyczyny RC2033.

Opcja GMWI jest używana z opcją GMWT. Opcja ta jest ignorowana, jeśli ta opcja nie jest określona. Jeśli zostanie podana, wartość GMWI musi być większa lub równa zero lub następująca wartość specjalna:

WIULIM

Nieograniczony przedział czasu oczekiwania.

Wartością początkową tego pola jest 0.

Wartości początkowe

Tabela 145. Początkowe wartości pól w MQGMO		
Nazwa pola	Nazwa stałej	Wartość stałej
GMSID	GMSIDV	'GMO¬'
GMVER	GMVER1	1
GMOPT	GMNWT	0
GMWI	Brak	0
GMSG1	Brak	0
GMSG2	Brak	0
GMRQN	Brak	Puste
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG	'¬'
GMSST	SSNSEG	'¬'
GMSEG	SEGIHB	'¬'
GMRE1	Brak	'¬'
GMTOK	MTKNON	Wartości null
GMRL	RLUNDF	-1
GMRS2	Brak	'¬'
GMMH	HMNONE	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..  
D*  
D* MQGMO Structure  
D*  
D* Structure identifier
```

```

D  GMSID                1      4      INZ('GMO ')
D* Structure version number
D  GMVER                5      8I 0 INZ(1)
D* Options that control the action of MQGET
D  GMOPT                9      12I 0 INZ(0)
D* Wait interval
D  GMWI                13     16I 0 INZ(0)
D* Signal
D  GMSG1               17     20I 0 INZ(0)
D* Signal identifier
D  GMSG2               21     24I 0 INZ(0)
D* Resolved name of destination queue
D  GMRQN               25     72      INZ
D* Options controlling selection criteria used for MQGET
D  GMMO               73     76I 0 INZ(3)
D* Flag indicating whether message retrieved is in a group
D  GMSGST             77     77      INZ(' ')
D* Flag indicating whether message retrieved is a segment of a
D* logical message
D  GMSST             78     78      INZ(' ')
D* Flag indicating whether further segmentation is allowed for the message
D* retrieved
D  GMSEG             79     79      INZ(' ')
D* Reserved
D  GMRE1             80     80      INZ
D* Message token
D  GMTOK             81     96      INZ('X'0000000000000000-
D                                0000000000000000')
D* Length of message data returned (bytes)
D  GMRL             97    100I 0 INZ(-1)
D* Reserved
D  GMRS2            101    104I 0 INZ(0)
D* Message handle
D  GMMH            105    112I 0 INZ(0)

```

MQIIH-nagłówek informacji IMS

Struktura MQIIH opisuje informacje, które muszą być obecne na początku komunikatu wysyłanego do mostu IMS za pomocą programu IBM MQ for z/OS.

Przegląd

Nazwa formatu: FMIMS.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQIIH i danych komunikatu aplikacji:

- Aplikacje, które łączą się z menedżerem kolejek, do którego należy kolejka mostu IMS, muszą udostępniać strukturę MQIIH, która znajduje się w zestawie znaków i kodowaniu menedżera kolejek. Wynika to z faktu, że konwersja danych struktury MQIIH nie jest wykonywana w tym przypadku.
- Aplikacje, które łączą się z innymi menedżerami kolejek, mogą udostępniać strukturę MQIIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań; konwersja tabeli MQIIH jest wykonywana przez odbierający agent kanału komunikatów połączony z menedżerem kolejek, który jest właścicielem kolejki mostu IMS.

Uwaga: Jest to tego jeden wyjątek. Jeśli menedżer kolejek, który jest właścicielem kolejki mostu IMS, używa produktu CICS do kolejkowania rozproszonego, tabela MQIIH musi znajdować się w zestawie znaków i kodowaniu menedżera kolejek, do którego należy kolejka mostu IMS.

- Dane komunikatu aplikacji zgodnie ze strukturą MQIIH muszą być w tym samym zestawie znaków i kodowaniu co struktura MQIIH. Pola *IICSI* i *IIENC* w strukturze MQIIH nie mogą być używane do określania zestawu znaków i kodowania danych komunikatu aplikacji.

Jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek, musi zostać dostarczone przez użytkownika wyjście konwersji danych w celu przekształcenia danych komunikatu aplikacji.

- [“Uwierzytelnianie kodów passtickets dla aplikacji pomostowych IMS” na stronie 1113](#)
- [“Pola” na stronie 1113](#)

- [“Wartości początkowe” na stronie 1116](#)
- [“Deklaracja RPG” na stronie 1116](#)

Uwierzytelnianie kodów passtickets dla aplikacji pomostowych IMS

Teraz administratorzy produktu IBM MQ mogą określić nazwę aplikacji, która ma być używana do uwierzytelniania kodów passtickets, dla aplikacji pomostowych IMS. W tym celu nazwa aplikacji jest określona jako nowy atrybut PTKTAPPL dla definicji obiektu STGCLASS, jako łańcuch alfanumeryczny o długości od 1 do 8 znaków.

Pusta wartość oznacza, że uwierzytelnianie odbywa się tak, jak w przypadku poprzednich wersji produktu IBM MQ, to znaczy nie ma przepływów nazw aplikacji dla żądania uwierzytelniania i zamiast niej używana jest wartość MVSxxxx.

Wartość z zakresu od 1 do 8 znaków alfanumerycznych musi być zgodna z regułami nazw aplikacji passticket zgodnie z opisem w publikacjach produktu RACF.

Administratorzy produktu IBM MQ i administratorzy produktu RACF muszą uzgodnić poprawne nazwy aplikacji, które mają być używane. Administrator produktu RACF musi utworzyć profil w klasie PTKTDATA z dostępem do odczytu identyfikatorów użytkowników wszystkich aplikacji, do których ma zostać przyznany dostęp. Administrator produktu IBM MQ musi utworzyć lub zmienić wymagane definicje STGCLASS, które określają nazwę aplikacji, która ma być używana na potrzeby uwierzytelniania przepustek.

Więcej informacji na ten temat zawiera publikacja *Script (MQSC) Command Reference*.

Pola

Struktura MQIIH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

IIAUT (8-bajtowy łańcuch znaków)

RACF (hasło lub passticket).

Jest to opcjonalne. Jeśli zostanie podany, jest on używany z identyfikatorem użytkownika w kontekście zabezpieczeń MQMD w celu zbudowania znacznika UTOKEN, który jest wysyłany do produktu IMS w celu udostępnienia kontekstu zabezpieczeń. Jeśli nie zostanie podany, identyfikator użytkownika zostanie użyty bez weryfikacji. Zależy to od ustawienia przełączników RACF, które mogą wymagać obecności elementu uwierzytelniającego.

Opcja ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL. Można użyć następującej wartości specjalnej:

IAUNON

Brak uwierzytelniania.

Długość tego pola jest podana przez LNAUTH. Wartością początkową tego pola jest IAUNON.

IICMT (łańcuch znakowy 1-bajtowy)

Tryb kontroli transakcji.

Więcej informacji na temat trybów zatwierdzania produktu IMS zawiera publikacja *OTMA Reference*. Wartość musi być jedną z następujących wartości:

ICMCTS

Zatwierdź następnie wyślij.

Ten tryb implikuje podwójne kolejkowanie danych wyjściowych, ale krótsze czasy zajętości regionu. Transakcje typu fast-path i conversational nie mogą być uruchamiane z tym trybem.

ICMSTC

Wyślij następnie zatwierdzenie.

Wartością początkową tego pola jest ICMCTS.

IICSI (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

IIENC (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest 0.

IIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Wartość musi być następująca:

IINONE

Brak flag.

Wartością początkową tego pola jest IINONE.

IIFMT (8-bajtowy łańcuch znaków)

IBM MQ nazwa formatu danych, które są następujące: MQIIH.

Określa nazwę formatu IBM MQ danych, które są zgodne ze strukturą MQIIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

IILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQIIH.

Wartość musi być następująca:

IILEN1

Długość struktury nagłówka informacji IMS .

Początkowa wartość tego pola to IILEN1.

IILTO (8-bajtowy łańcuch znaków)

Nadpisanie terminalu logicznego.

Pole to jest umieszczane w polu PCB we/wy. Jest ona opcjonalna. Jeśli nie zostanie podana nazwa potoku TPIPE, zostanie użyty. Wartość ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL.

Długość tego pola jest podana przez LNLTOV. Początkowa wartość tego pola to 8 znaków odstępu.

IIMMN (8-bajtowy łańcuch znaków)

Nazwa odwzorowania usług formatu komunikatu.

Pole to jest umieszczane w polu PCB we/wy. Jest ono opcjonalne. Na wejściu reprezentuje identyfikator MID, na wyjściu reprezentuje on MOD. Wartość ta jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL.

Długość tego pola jest podana przez LNMFMN. Początkowa wartość tego pola to 8 znaków odstępu.

IIRFM (8-bajtowy łańcuch znaków)

IBM MQ -nazwa formatu komunikatu odpowiedzi.

Jest to nazwa formatu produktu IBM MQ komunikatu odpowiedzi, który zostanie wysłany w odpowiedzi na bieżący komunikat. Reguły kodowania są takie same jak w przypadku pola *MDFMT* w strukturze MQMD.

Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

IIRSV (łańcuch znakowy 1-bajtowy)

Zarezerwowane.

Jest to pole zastrzeżone. Musi być puste.

IISEC (łańcuch znakowy 1-bajtowy)

Zasięg zabezpieczeń.

Oznacza to, że wymagane jest przetwarzanie zabezpieczeń produktu IMS . Zdefiniowane są następujące wartości:

ISSCHK

Sprawdź zasięg zabezpieczeń.

ACEE jest budowany w regionie sterującym, ale nie w regionie zależnym.

ISSFUL

Pełny zasięg zabezpieczeń.

Buforowany ACEE jest budowany w regionie sterującym, a niebuforowany ACEE jest budowany w regionie zależnym. Jeśli używany jest system ISSFUL, należy upewnić się, że ID użytkownika, dla którego zbudowano ACEE, ma dostęp do zasobów używanych w regionie zależnym.

Jeśli dla tego pola nie określono ISSCHK i ISSFUL, przyjmuje się, że ISSCHK.

Wartością początkową tego pola jest ISSCHK.

IISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

IISIDV

Identyfikator struktury nagłówka informacji IMS .

Wartością początkową tego pola jest IISIDV.

IITID (16-bajtowy łańcuch bitowy)

Identyfikator instancji transakcji.

To pole jest używane przez komunikaty wyjściowe z produktu IMS , dlatego jest ignorowane przy pierwszym wejściu. Jeśli parametr *IITST* jest ustawiony na ITSIC, to musi on być podany w następnym wejściu i wszystkie kolejne wejścia, aby umożliwić IMS korelowanie komunikatów w poprawnej konwersacji. Można użyć następującej wartości specjalnej:

ITINON

Brak identyfikatora instancji transakcji.

Długość tego pola jest podana przez LNTIID. Wartością początkową tego pola jest ITINON.

IITST (łańcuch znakowy 1-bajtowy)

Stan transakcji.

Wskazuje stan konwersacji IMS . Ta opcja jest ignorowana przy pierwszym wejściu, ponieważ żadna konwersacja nie istnieje. W kolejnych danych wejściowych wskazuje, czy konwersacja jest aktywna, czy nie. W przypadku wyjścia jest on ustawiany przez produkt IMS. Wartość musi być jedną z następujących wartości:

ITSIC

W rozmowie.

ITSNIC

Nie w rozmowie.

ITSARC

Zwróć dane stanu transakcji w postaci architected.

Ta wartość jest używana tylko z komendą IMS /DISPLAY TRAN . Powoduje to, że dane stanu transakcji są zwracane w postaci architected IMS zamiast postaci znaku. Więcej informacji na ten temat zawiera sekcja Zapisywanie programów transakcyjnych IMS za pomocą programu IBM MQ .

Wartością początkową tego pola jest ITSNIC.

IIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

IIVER1

Numer wersji dla struktury nagłówka informacji IMS .

Następująca stała określa numer wersji bieżącej wersji:

IIVERC

Bieżąca wersja struktury nagłówka informacyjnego produktu IMS .

Początkowa wartość tego pola to IIVER1.

Wartości początkowe

Tabela 146. Początkowe wartości pól w MQIIH		
Nazwa pola	Nazwa stałej	Wartość stałej
IISID	IISIDV	'IIH¬'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	Brak	0
IICSI	Brak	0
IIFMT	FMNONE	Puste
IIFLG	IINONE	0
IILTO	Brak	Puste
IIMMN	Brak	Puste
IIRFM	FMNONE	Puste
IIAUT	IAUNON	Puste
IITID	ITINON	Wartości null
IITST	ITSNIC	'¬'
IICMT	ICMCTS	'0'
IISEC	ISSCHK	'C'
IIRSV	Brak	'¬'

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQIIH Structure
```



```

D*
D* Structure identifier
D IISID 1 4 INZ('IIH ')
D* Structure version number
D IIVER 5 8I 0 INZ(1)
D* Length of MQIIH structure
D IILEN 9 12I 0 INZ(84)
D* Reserved
D IIENC 13 16I 0 INZ(0)
D* Reserved
D IICSI 17 20I 0 INZ(0)
D* MQ format name of data that followsMQIIH
D IIFMT 21 28 INZ(' ')
D* Flags
D IIFLG 29 32I 0 INZ(0)
D* Logical terminal override
D IILTO 33 40 INZ
D* Message format services map name
D IIMMN 41 48 INZ
D* MQ format name of reply message
D IIRFM 49 56 INZ(' ')
D* RACF password or passticket
D IIAUT 57 64 INZ(' ')
D* Transaction instance identifier
D IITID 65 80 INZ(X'0000000000000000-
0000000000000000')
D
D* Transaction state
D IITST 81 81 INZ(' ')
D* Commit mode
D IICMT 82 82 INZ('0')
D* Security scope
D IISEC 83 83 INZ('C')
D* Reserved
D IIRSV 84 84 INZ

```

MQIMPO-Zapytanie o opcje właściwości komunikatu

Struktura MQIMPO umożliwia aplikacjom określanie opcji, które sterują sposobem uzyskiwania informacji o właściwościach komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym w wywołaniu MQINQMP.

Zestaw znaków i kodowanie: Dane w tabeli MQIMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1117](#)
- [“Wartości początkowe” na stronie 1123](#)
- [“Deklaracja RPG” na stronie 1123](#)

Pola

Struktura MQIMPO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

IPOPT (10-cyfrowa liczba całkowita ze znakiem)

Następujące opcje sterują działaniem komendy MQINQMP. Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe). Należy zauważyć, że kombinacje opcji, które nie są poprawne, są poprawne. Wszystkie pozostałe kombinacje są poprawne.

Opcje danych wartości: Następujące opcje odnoszą się do przetwarzania danych wartości, gdy właściwość jest pobierana z komunikatu.

IPCVAL

Ta opcja żąda, aby wartość właściwości została przekształcona w taki sposób, aby była zgodna z wartościami *IPREQCSI* i *IPREQENC* określonymi przed wywołaniem wywołania MQINQMP, zwracając wartość właściwości w obszarze *Value*.

- Jeśli konwersja powiedzie się, pola *IPRETCSI* i *IPRETENC* są ustawione na takie same, jak *IPREQCSI* i *IPREQENC* po powrocie z wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wartość właściwości zostanie zwrócona bez konwersji.

Jeśli właściwość jest łańcuchem, pola *IPRETCSI* i *IPRETENC* są ustawiane na zestaw znaków i kodowanie nieprzekształconego łańcucha.

Kod zakończenia to CCWARN w tym przypadku z kodem przyczyny RC2466. Cursor właściwości jest zaawansowany do zwróconej właściwości.

Jeśli wartość właściwości zostanie rozwinięta podczas konwersji, i przekracza wielkość parametru *Value*, zwracana jest wartość nieprzekształcona, a kod zakończenia CCFAIL; kod przyczyny jest ustawiony na RC2469.

Parametr *DataLength* wywołania MQINQMP zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmieniony.

Ta opcja wymaga również, aby:

- Jeśli nazwa właściwości zawiera znak wieloznaczny, oraz
- Pole *IPRETNAMECHRP* jest inicjowane za pomocą adresu lub przesunięcia dla zwróconej nazwy,

Zwracana nazwa jest przekształcana w taki sposób, aby była zgodna z wartościami *IPREQCSI* i *IPREQENC*.

- Jeśli konwersja się powiedzie, pole *VSCCSID* produktu *IPRETNAMECHRP* i kodowanie zwróconej nazwy są ustawiane na wartość wejściową *IPREQCSI* i *IPREQENC*.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu lub ostrzeżenia, zwracana nazwa jest nieprzekształcona. Kod zakończenia to CCWARN w tym przypadku z kodem przyczyny RC2492.

Cursor właściwości jest zaawansowany do zwróconej właściwości. Wartość RC2466 jest zwracana, jeśli nie została przekształcona wartość i nazwa.

Jeśli zwrócona nazwa zostanie rozwinięta podczas konwersji i zostanie przekroczona wielkość pola *VSBufsize* w *RequestedName*, zwrócony łańcuch nie zostanie przekształcony, a kod zakończenia CCFAIL i kod przyczyny są ustawione na RC2465.

Pole *VSLength* struktury MQCHARV zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmieniony.

IPCTYP

Ta opcja żąda, aby wartość właściwości została przekształcona z jej bieżącego typu danych w typ danych określony w parametrze *Type* wywołania MQINQMP.

- Jeśli konwersja powiedzie się, parametr *Type* nie zostanie zmieniony podczas powrotu wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP w inny sposób zakończy się bez błędu, wywołanie zakończy się niepowodzeniem z powodu RC2470. Cursor właściwości jest niezmieniony.

Jeśli konwersja typu danych powoduje rozwinięcie wartości podczas konwersji, a wartość przekształcona przekracza wielkość parametru *Value*, zwracana jest wartość bez konwersji, a kod zakończenia CCFAIL i kod przyczyny są ustawione na wartość RC2469.

Parametr *DataLength* wywołania MQINQMP zwraca długość, do której została przekształcona wartość właściwości w celu umożliwienia aplikacji określenia wielkości buforu wymaganego do uwzględnienia przekształconej wartości właściwości. Cursor właściwości jest niezmieniony.

Jeśli wartość parametru *Type* wywołania MQINQMP nie jest poprawna, wywołanie kończy się niepowodzeniem z powodu RC2473.

Jeśli żądana konwersja typu danych nie jest obsługiwana, wywołanie kończy się niepowodzeniem z powodu RC2470. Obsługiwane są następujące konwersje typów danych:

Typ danych właściwości	Obsługiwane docelowe typy danych
TYPBOL	TYPSTR, TYPI8, TYPI16, TYPI32, TYPI64
TYPBST	TYPSTR
TYPI8	TYPSTR, TYPI16, TYPI32, TYPI64
TYPI16	TYPSTR, TYPI32, TYPI64
TYPI32	TYPSTR, TYPI64
TYPI64	TYPSTR
TYPF32	TYPSTR, TYPF64
TYPF64	TYPSTR
TYPSTR	TYPBOL, TYPI8, TYPI16, TYPI32, TYPI64, TYPF32, TYPF64
TYPNUL	Brak

Ogólne reguły dotyczące obsługiwanych konwersji są następujące:

- Wartości właściwości liczbowych mogą być przekształcane z jednego typu danych na inny, pod warunkiem że w trakcie konwersji nie zostaną utracone żadne dane.

Na przykład wartość właściwości o typie danych TYPI32 może zostać przekształcona w wartość o typie danych TYPI64, ale nie może zostać przekształcona w wartość o typie danych TYPI16.

- Wartość właściwości dowolnego typu danych może zostać przekształcona w łańcuch.
- Wartość właściwości łańcuchowej może zostać przekształcona w dowolny inny typ danych pod warunkiem, że łańcuch zostanie poprawnie sformatowany w celu konwersji. Jeśli aplikacja podejmie próbę przekształcenia wartości właściwości łańcuchowej, która nie jest poprawnie sformatowana, program IBM MQ zwróci kod przyczyny RC2472.
- Jeśli aplikacja podejmie próbę konwersji, która nie jest obsługiwana, program IBM MQ zwróci kod przyczyny RC2470.

Szczegółowe reguły przekształcania wartości właściwości z jednego typu danych na inny są następujące:

- W przypadku konwersji wartości właściwości TYPBOL na łańcuch wartość TRUE jest przekształcana w łańcuch "TRUE", a wartość false jest przekształcana w łańcuch "FALSE".
- Podczas konwersji wartości właściwości TYPBOL na liczbowy typ danych wartość TRUE jest przekształcana na wartość 1, a wartość FALSE jest przekształcana na zero.
- Podczas konwersji wartości właściwości łańcuchowej na wartość TYPBOL, łańcuch "TRUE" lub "1" jest przekształcany na TRUE, a łańcuch "FALSE" lub "0" jest przekształcany na wartość FALSE.

Należy zauważyć, że wielkość liter "TRUE" i "FALSE" nie jest rozróżniana.

Żaden inny łańcuch nie może zostać przekształcony; IBM MQ zwraca kod przyczyny RC2472.

- Podczas przekształcania wartości właściwości łańcuchowej na wartość o typie danych TYPI8, TYPI16, TYPI32 lub TYPI64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits
```

Znaczenia składników tego łańcucha są następujące:

blanks

Opcjonalne wiodące puste znaki

sign

Opcjonalny znak plus (+) lub znak minus (-).

digits

Ciągła sekwencja znaków cyfr (0-9). Musi istnieć co najmniej jeden znak cyfry.

Po sekwencji znaków cyfr łańcuch może zawierać inne znaki, które nie są znakami cyfr, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę całkowitą.

IBM MQ zwraca kod przyczyny RC2472 , jeśli łańcuch nie jest poprawnie sformatowany.

- W przypadku konwersji wartości właściwości łańcuchowej na wartość o typie danych TYPF32 lub TYPF64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

Znaczenia składników tego łańcucha są następujące:

blanks

Opcjonalne wiodące puste znaki

sign

Opcjonalny znak plus (+) lub znak minus (-).

digits

Ciągła sekwencja znaków cyfr (0-9). Musi istnieć co najmniej jeden znak cyfry.

e_char

Znak wykładnika, który jest albo "E", albo "e".

e_sign

Opcjonalny znak plus (+) lub znak minus (-) dla wykładnika.

e_digits

Ciągła sekwencja znaków cyfr (0-9) dla wykładnika. Jeśli łańcuch zawiera znak wykładnika, musi być obecny co najmniej jeden znak cyfry.

Po sekwencji znaków cyfr lub opcjonalnych znaków reprezentujących wykładnik, łańcuch może zawierać inne znaki, które nie są znakami cyfr, ale konwersja zatrzymuje się, gdy tylko pierwszy z tych znaków zostanie osiągnięty. Przyjmuje się, że łańcuch reprezentuje liczbę dziesiętną zmiennopozycyjną z wykładnikiem, który jest potęgą liczbą 10.

IBM MQ zwraca kod przyczyny RC2472 , jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania wartości liczbowej właściwości w łańcuch wartość jest przekształcana na łańcuchową reprezentację wartości jako liczbę dziesiętną, a nie łańcuchową zawierającą znak ASCII dla tej wartości. Na przykład liczba całkowita 65 jest przekształcana w łańcuch "65", a nie łańcuch "A".
- Podczas przekształcania wartości właściwości łańcucha bajtowego w łańcuch każdy bajt jest przekształcany w dwa znaki szesnastkowe, które reprezentują bajt. Na przykład tablica bajtów {0xF1, 0x12, 0x00, 0xFF} jest przekształcana w łańcuch "F11200FF".

IPQLEN

Zapytanie o typ i długość wartości właściwości. Długość jest zwracana w parametrze *DataLength* wywołania MQINQMP. Wartość właściwości nie jest zwracana.

Jeśli zostanie podany bufor *ReturnedName* , to pole *VSLength* struktury MQCHARV zostanie wypełnione nazwą właściwości. Nazwa właściwości nie jest zwracana.

Opcje iteracji: Następujące opcje są powiązane z iteracją nad właściwościami przy użyciu nazwy ze znakiem wieloznacznym

IPINQF

Sprawdź pierwszą właściwość, która jest zgodna z podaną nazwą. Po wywołaniu tej operacji na obiekcie, który jest zwracany, zostanie utworzony kursor.

Jest to wartość domyślna.

Następnie można użyć opcji IPINQC z wywołaniem MQINQMP, jeśli jest to wymagane, aby ponownie uzyskać informacje na temat tej samej właściwości.

Należy zauważyć, że istnieje tylko jeden kursor właściwości, dlatego jeśli nazwa właściwości określona w wywołaniu MQINQMP, zmienia kursor, zostanie zresetowana.

Ta opcja nie jest poprawna z jedną z następujących opcji:

IPINQN

IPINQC

IPINQN

Sprawdź następną właściwość, która jest zgodna z podaną nazwą, kontynuując wyszukiwanie z kursora właściwości. Kursor jest awansowany do zwróconej właściwości.

Jeśli jest to pierwsze wywołanie MQINQMP dla podanej nazwy, zwracana jest pierwsza właściwość, która jest zgodna z podaną nazwą.

Następnie można użyć opcji IPINQC z wywołaniem MQINQMP, jeśli jest to wymagane, aby ponownie uzyskać informacje na temat tej samej właściwości.

Jeśli właściwość pod kursorem została usunięta, komenda MQINQMP zwraca następną zgodną właściwość po usunięciu tej właściwości.

Jeśli właściwość zostanie dodana, która jest zgodna ze znakiem wieloznacznym, podczas gdy iteracja jest w toku, ta właściwość może lub nie może zostać zwrócona podczas kończenia iteracji. Ta właściwość jest zwracana po restarcie iteracji przy użyciu IPINQF.

Właściwość pasująca do znaku wieloznacznego, który został usunięty, podczas gdy iteracja była w toku, nie jest zwracana po jego usunięciu.

Ta opcja nie jest poprawna z jedną z następujących opcji:

IPINQF

IPINQC

IPINQC

Pobieranie wartości właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana, przy użyciu opcji IPINQF lub IPINQN.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany, gdy uchwyt komunikatu jest określony w polu *MsgHandle* obiektu MQGMO w wywołaniu MQGET, lub gdy uchwyt komunikatu jest określony w polach *OriginalMsgHandle* lub *NewMsgHandle* struktury MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i z przyczyną RC2471.

Ta opcja nie jest poprawna z jedną z następujących opcji:

IPINQF

IPINQN

Jeśli żadna z wcześniej opisanych opcji nie jest wymagana, można użyć następującej opcji:

IPNONE.

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

IPNONE jest pomocna w dokumentacji programu; nie jest zamierzone, aby ta opcja była używana z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest IPINQF.

IPREQCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków, w którym wartość właściwości *inquired* ma zostać przekształcona w łańcuch znaków, jeśli wartość ta jest łańcuchem znaków. Jest to również zestaw znaków, w którym ma zostać przekształcona *ReturnedName*, jeśli określono IPCVAL lub IPCTYP.

Wartością początkową tego pola jest CSAPL.

IPREQENC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kodowanie, w którym wartość właściwości *inquired* ma zostać przekształcona w przypadku określenia wartości IPCVAL lub IPCTYP.

Wartością początkową tego pola jest ENNAT.

IPRE1 (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest pustym znakiem.

IPRETCSI (10-cyfrowa liczba całkowita ze znakiem)

W przypadku danych wyjściowych jest to zestaw znaków wartości zwracanej, jeśli parametr *Type* wywołania MQINQMP to TYPSTR.

Jeśli podano opcję IPCVAL, a konwersja się powiodła, pole *ReturnedCCSID* (w zamian) ma taką samą wartość, jak wartość przekazana.

Początkowa wartość tego pola wynosi zero.

IPRETENC (10-cyfrowa liczba całkowita ze znakiem)

W przypadku danych wyjściowych jest to kodowanie zwracanej wartości.

Jeśli podano opcję IPCVAL, a konwersja się powiodła, pole *ReturnedEncoding* (w zamian) ma taką samą wartość, jak wartość przekazana.

Wartością początkową tego pola jest ENNAT.

IPRETNAMCHRP (10-cyfrowa liczba całkowita ze znakiem)

Rzeczywista nazwa właściwości zapytania.

W przypadku wejścia bufor łańcuchowy może być przekazywany za pomocą pola *VSPtr* lub *VSOffset* struktury MQCHARV. Długość buforu łańcucha jest określona za pomocą pola *VSBufsize* struktury MQCHARV.

W przypadku powrotu z wywołania MQINQMP bufor łańcucha jest uzupełniany nazwą właściwości, która została zapytana, pod warunkiem, że bufor łańcuchowy był wystarczająco długi, aby mógł w pełni zawierać nazwę. Pole *VSLength* struktury MQCHARV jest wypełnione przez długość nazwy właściwości. Pole *VSCCSID* struktury MQCHARV jest wypełniane w celu wskazania zestawu znaków zwracanej nazwy, bez względu na to, czy konwersja nazwy nie powiodła się.

Jest to pole wejściowe/wyjściowe. Wartością początkową tego pola jest MQCHARV_DEFAULT.

IPSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

IPSIDV

Identyfikator zapytania o strukturę opcji właściwości komunikatu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest IPSIDV.

IPTYP (10-cyfrowa liczba całkowita ze znakiem)

Reprezentacja łańcuchowa typu danych właściwości.

Jeśli właściwość została określona w nagłówku MQRFH2, a atrybut MQRFH2 dt nie został rozpoznany, to pole może zostać użyte do określenia typu danych właściwości. *TypeString* jest zwracany w kodowanym zestawie znaków 1208 (UTF-8) i jest to pierwsze osiem bajtów wartości atrybutu dt właściwości, które nie zostały rozpoznane.

To jest zawsze pole wyjściowe. Wartość początkowa tego pola jest łańcuchem pustym w języku programowania C, a 8 znaków odstępu w innych językach programowania.

IPVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

IPVER1

Numer wersji dla zapytania o strukturę opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

IPVERC

Bieżąca wersja struktury opcji sprawdzania właściwości komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to IPVER1.

Wartości początkowe

Tabela 147. Początkowe wartości pól w MQIPMO		
Nazwa pola	Nazwa stałej	Wartość stałej
IPSID	IPSIDV	'IMPO'
IPVER	IPVER1	1
IPOPT	IPINQF	
IPREQENC	ENNAT	
IPREQCSI	CSAPL	
IPRETENC	ENNAT	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		wartości puste

Deklaracja RPG

```
D* MQIMPO Structure
D*
D*
D* Structure identifier
```

```

D IPSID          1   4  INZ('IMPO')
D*
D* Structure version number
D IPVER          5   8I 0 INZ(1)
D*
D* Options that control the action of
D* MQINQMP
D IPOPT         9  12I 0 INZ(0)
D*
D* Requested encoding of Value
D IPREQENC      13  16I 0 INZ(273)
D*
D* Requested character set identifier
D* of Value
D IPREQCSI      17  20I 0 INZ(-3)
D*
D* Returned encoding of Value
D IPRETENC      21  24I 0 INZ(273)
D*
D* Returned character set identifier of
D* Value
D IPRETCSI      25  28I 0 INZ(0)
D*
D* Reserved
D IPRE1         29  32I 0 INZ(0)
D*
D* Returned property name
D* Address of variable length string
D IPRETNAMCHRP  33  48* INZ(*NULL)
D* Offset of variable length string
D IPRETNAMCHRO  49  52I 0 INZ(0)
D* Size of buffer
D IPRETNAMVSBS  53  56I 0 INZ(-1)
D* Length of variable length string
D IPRETNAMCHRL  57  60I 0 INZ(0)
D* CCSID of variable length string
D IPRETNAMCHRC  61  64I 0 INZ(-3)
D*
D* Property data type as a string
D IPTYP        65  72  INZ

```

MQMD-deskryptor komunikatu

Przegląd

Cel: Struktura MQMD zawiera informacje sterujące, które towarzyszą danym aplikacji podczas przesyłania komunikatów między aplikacjami wysyłającym i odbierającym. Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQGET, MQPUT i MQPUT1.

Wersja: Bieżąca wersja deskryptora MQMD to MDVER2. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję deskryptora MQMD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *MDVER* ustawioną na MDVER1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *MDVER* na numer wersji wymaganej wersji.

Deklaracja dla struktury version-1 jest dostępna z nazwą MQMD1.

Zestaw znaków i kodowanie: Dane w strukturze MQMD muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako IBM MQ MQI client, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Jeśli menedżery kolejek wysyłających i odbierających korzystają z różnych zestawów znaków lub kodowań, dane w strukturze MQMD są przekształcane automatycznie. Nie jest konieczne, aby aplikacja przekształciła deskryptor MQMD.

- [“Korzystanie z różnych wersji deskryptora MQMD” na stronie 1125](#)
- [“Kontekst komunikatu” na stronie 1125](#)
- [“Utrata ważności komunikatu” na stronie 1126](#)

- [“Pola” na stronie 1126](#)
- [“Wartości początkowe” na stronie 1168](#)
- [“Deklaracja RPG” na stronie 1169](#)

Korzystanie z różnych wersji deskryptora MQMD

Deskryptor MQMD w wersji version-2 jest na ogół równoważny z użyciem deskryptora MQMD z wersji version-1 i z przedrostkiem danych komunikatu z strukturą MQMDE. Jeśli jednak wszystkie pola w strukturze MQMDE mają swoje wartości domyślne, można pominąć MQMDE. W dalszej części tej sekcji używane są następujące elementy: version-1 MQMD plus MQMDE.

- W wywołaniach MQPUT i MQPUT1, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, aplikacja może opcjonalnie prefikować dane komunikatu za pomocą MQMDE, ustawiając pole *MDFMT* w MQMD na FMMDE, aby wskazać, że jest obecna MQMDE. Jeśli aplikacja nie udostępnia wywołania MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w MQMDE.

Uwaga: Kilka pól istniejących w strukturze MQMD version-2, ale nie version-1 MQMD, są polami wejściowymi/wyjściowymi w wywołaniach MQPUT i MQPUT1. Jednak menedżer kolejek *nie* zwraca żadnych wartości w równoważnych polach w MQMDE na danych wyjściowych z wywołań MQPUT i MQPUT1. Jeśli aplikacja wymaga tych wartości wyjściowych, musi ona używać deskryptora MQMD z wersji version-2.

- W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, menedżer kolejek prefiksuje komunikat zwrócony przez produkt MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Pole *MDFMT* w strukturze MQMD będzie miało wartość FMMDE, aby wskazać, że jest obecna MQMDE.

Wartości domyślne, które menedżer kolejek używany dla pól w MQMDE są takie same, jak początkowe wartości tych pól, wyświetlane w [Tabela 148 na stronie 1168](#).

Jeśli komunikat znajduje się w kolejce transmisji, niektóre pola w strukturze MQMD są ustawiane na określone wartości. Szczegółowe informacje zawiera sekcja [“MQXQH-nagłówków kolejki transmisji” na stronie 1265](#).

Kontekst komunikatu

Niektóre pola w strukturze MQMD zawierają kontekst komunikatu. Zwykle:

- Kontekst **Tożsamość** odnosi się do aplikacji, która *pierwotnie* umieszczała komunikat
- Kontekst **Pochodzenie** odnosi się do aplikacji, która *ostatnio* umieszczała komunikat.
- Kontekst **Użytkownik** odnosi się do aplikacji, która *oryginalnie* umieściła komunikat.

Te dwie aplikacje mogą być tą samą aplikacją, ale mogą to być także różne aplikacje (na przykład, gdy komunikat jest przekazywany z jednej aplikacji do innej).

Chociaż kontekst tożsamości i pochodzenia zwykle ma znaczenie opisane wcześniej, treść obu typów pól kontekstu w strukturze MQMD zależy faktycznie od opcji PM*, które są określone podczas umieszczania komunikatu. W rezultacie kontekst tożsamości nie musi być powiązany z aplikacją, która pierwotnie umieściła komunikat, a kontekst źródłowy niekoniecznie odnosi się do aplikacji, która ostatnio wstawiła komunikat-zależy to od projektu pakietu aplikacji.

Istnieje jedna klasa aplikacji, która nigdy nie zmienia kontekstu komunikatu, a mianowicie agenta kanału komunikatów (MCA). MCAs odbierający komunikaty ze zdalnych menedżerów kolejek korzysta z opcji kontekstu PMSETA w wywołaniu MQPUT lub MQPUT1. Pozwala to odbierającym agentowi MCA na zachowanie dokładnie kontekstu komunikatu, który podróżował z komunikatem od wysyłającego agenta MCA. Jednak wynikiem jest, że kontekst źródłowy nie odnosi się do aplikacji, która ostatnio umieściła komunikat (odbierający MCA), ale zamiast tego odnosi się do wcześniejszej aplikacji, która umieściła komunikat (prawdopodobnie sama aplikacja źródłowa).

Więcej informacji na ten temat zawiera sekcja [Kontekst komunikatu](#).

Utrata ważności komunikatu

Komunikaty, które utraciły ważność w załadowanej kolejce (kolejka, która została otwarta), są automatycznie usuwane z kolejki w rozsądnym czasie po ich upływie. Niektóre inne nowe funkcje tej wersji produktu IBM MQ mogą prowadzić do skanowania załadowanych kolejek rzadziej niż w poprzedniej wersji produktu, jednak wygasłe komunikaty w załadowanych kolejkach są zawsze usuwane w rozsądnym okresie ich utraty ważności.

Pola

Struktura MQMD zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

MDACC (32-bajtowy łańcuch bitowy)

Token rozliczania.

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Produkt MDACC umożliwia aplikacji wykonanie pracy wykonanej w wyniku komunikatu, który ma być odpowiednio obciążony. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza jego zawartości.

Gdy menedżer kolejek generuje te informacje, jest on ustawiany w następujący sposób:

- Pierwszy bajt pola jest ustawiony na długość informacji rozliczeniowych znajdujących się w następujących bajtach. Długość ta mieści się w zakresie od zera do 30 i jest przechowywana w pierwszym bajcie jako binarna liczba całkowita.
- Drugi i kolejne bajty (określone w polu długości) są ustawiane na informacje rozliczeniowe odpowiednie dla środowiska.
 - W systemie z/OS informacje rozliczeniowe są ustawiane na:
 - W przypadku zadania wsadowego z/OS -informacje rozliczeniowe z karty JES JOB lub z instrukcji JES ACCT na karcie EXEC (separatory przecinków są zmieniane na X'FF '). Informacje te są obcinane, jeśli to konieczne, do 31 bajtów.
 - Dla TSO, numer konta użytkownika.
 - W przypadku systemu CICSjednostka logiczna (LU 6.2) identyfikatora pracy (UEPUOWDS) (26 bajtów).
 - W przypadku systemu IMS8-znakowa nazwa PSB konkatenowana z 16-znakowym znacznikiem odtwarzania IMS .
 - W systemie IBM iinformacje rozliczeniowe są ustawiane na kod rozliczeniowy dla zadania.
 - W systemach HP Integrity NonStop Serveri Systemy UNIXinformacje rozliczeniowe są ustawiane na liczbowy identyfikator użytkownika (w postaci znaków ASCII).
 - W systemie Windowsinformacje rozliczeniowe są ustawiane na identyfikator zabezpieczeń systemu Windows NT (SID) w formacie skompresowanym. Identyfikator SID jednoznacznie identyfikuje identyfikator użytkownika zapisany w polu MDUID . Jeśli identyfikator SID jest zapisany w polu MDACC , zostanie pominięty 6-bajtowy Urząd Identyfikatora (znajdujący się w trzecim i kolejnych bajtach identyfikatora SID). Na przykład, jeśli identyfikator SID Windows NT ma długość 28 bajtów, w polu MDACC zapisywane są 22 bajty informacji o identyfikatorze SID.
- Ostatni bajt jest ustawiony na typ znacznika rozliczania, jedną z następujących wartości:

ATTACK

CICS Identyfikator LUOW.

ATTDOS

Domyślny token rozliczania PC DOS.

ATTWNT

Identyfikator zabezpieczeń produktu Windows .

ATT400

IBM i token rozliczania.

ATTUNX

Identyfikator liczbowy Systemy UNIX .

ATTUSR

Zdefiniowany przez użytkownika token rozliczania.

ATTUNK

Nieznany typ znacznika rozliczania.

Typ znacznika rozliczania jest ustawiony na wartość jawną tylko w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Windows oraz IBM MQ MQI clients połączonych z tymi systemami. W innych środowiskach typ znacznika księgowego jest ustawiany na wartość ATTUNK. W tych środowiskach można użyć pola *MDPAT* , aby wynioskować typ odebranego znacznika rozliczeniowego.

- Wszystkie pozostałe bajty są ustawione na zero binarne.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PMO* określono PMSETI lub PMSETA. Jeśli nie określono ani PMSETI, ani PMSETA, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDACC* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDACC* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET w [“MQPMO-Put-message, opcje”](#) na stronie 1191 , aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest używana jako *MDACC* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDACC* we wszystkich publikacjach wysyłanych do tych subskrybentów. Jeśli komunikat nie ma kontekstu, to pole jest całkowicie binarne zero.

To jest pole wyjściowe dla wywołania MQGET.

To pole nie podlega żadnym tłumaczeniom opartym na zestawie znaków menedżera kolejek-pole jest traktowane jako łańcuch bitów, a nie jako łańcuch znaków.

Menedżer kolejek nie wykonuje żadnych informacji z informacjami w tym polu. Aplikacja musi interpretować informacje, jeśli chce korzystać z informacji do celów księgowych.

W polu *MDACC* można użyć następującej wartości specjalnej:

ACNONE

Nie określono znacznika rozliczeniowego.

Wartość jest binarna zero dla długości pola.

Długość tego pola jest podana przez LNACT. Wartością początkową tego pola jest ACNONE.

MDAID (32-bajtowy łańcuch znaków)

Dane aplikacji odnoszące się do tożsamości.

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

MDAID to informacje, które są definiowane przez pakiet aplikacji i mogą być używane w celu udostępnienia dodatkowych informacji o komunikacie lub jego inicjatorze. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Gdy menedżer kolejek generuje te informacje, jest on całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PMO* określono PMSETI lub PMSETA. Jeśli występuje znak o kodzie zero, wartość NULL i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli nie określono ani PMSETI, ani PMSETA, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDAID*, który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDAID* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDAID*, gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDAID* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola nadawana jest przez LNAIDD. Początkowa wartość tego pola to 32 znaki puste.

MDAOD (4-bajtowy łańcuch znaków)

Dane aplikacji związane z pochodzeniem.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

MDAOD to informacje zdefiniowane przez pakiet aplikacji, które mogą być używane do udostępniania dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiona przez aplikacje działające z odpowiednim uprawnieniem użytkownika w celu wskazania, czy dane tożsamości są zaufane.

Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Gdy menedżer kolejek generuje te informacje, jest on całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr PMSETA jest określony w parametrze *PMO*. Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli wartość PMSETA nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDAOD*, który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDAOD* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDAOD*, gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDAOD* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola nadawana jest przez LNAORD. Początkowa wartość tego pola to 4 puste znaki.

MDBOC (10-cyfrowa liczba całkowita ze znakiem)

Licznik wycofań.

Jest to liczba określająca, ile razy komunikat został wcześniej zwrócony przez wywołanie MQGET jako część jednostki pracy, a następnie wycofał się z niego. Jest ona udostępniana jako pomoc dla aplikacji w wykrywaniu błędów przetwarzania, które są oparte na treści komunikatu. Liczba nie obejmuje wywołań MQGET, które określiły dowolną z opcji GMBRW*.

Dokładność tego licznika ma wpływ na atrybut kolejki *HardenGetBackout*; patrz [“Atrybuty dla kolejek” na stronie 1395](#).

To jest pole wyjściowe dla wywołania MQGET. Jest on ignorowany w przypadku wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest 0.

MDCID (24-bajtowy łańcuch bitowy)

Identyfikator korelacji.

Jest to łańcuch bajtowy, którego aplikacja może użyć do powiązania jednego komunikatu z innym, lub do powiązania komunikatu z innymi pracami wykonywanego przez aplikację. Identyfikator korelacji

jest stałą właściwością komunikatu i utrzymuje się między restartami menedżera kolejek. Ponieważ identyfikator korelacji jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator korelacji *nie* jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do drugiego.

W przypadku wywołań MQPUT i MQPUT1 aplikacja może określić dowolną wartość. Menedżer kolejek przesyła tę wartość wraz z komunikatem i dostarcza ją do aplikacji, która wysyła żądanie pobrania komunikatu.

Jeśli aplikacja określa identyfikator PMNCID, menedżer kolejek generuje unikalny identyfikator korelacji, który jest wysyłany z komunikatem, a także jest zwracany do aplikacji wysyłającej na wyjściu z wywołania MQPUT lub MQPUT1.

Ten wygenerowany identyfikator korelacji jest przechowywany razem z komunikatem, jeśli jest on zachowywany i jest używany jako identyfikator korelacji, gdy komunikat jest wysyłany jako publikacja do subskrybentów, którzy określają CINONE w polu SDCID w zmaterializowanej tabeli MQSD, która została przekazana w wywołaniu MQSUB.

Więcej informacji na temat zachowanych publikacji zawiera sekcja [“MQPMO-Put-message, opcje” na stronie 1191](#).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole MDCID w sposób określony przez pole MDREP oryginalnego komunikatu, ROCMTC lub ROPCI. Aplikacje, które generują komunikaty raportów, powinny również to robić.

W przypadku wywołania MQGET MDCID jest jednym z pięciu pól, których można użyć do wybrania konkretnego komunikatu, który ma zostać pobrany z kolejki. Aby uzyskać szczegółowe informacje na temat określania wartości dla tego pola, należy zapoznać się z opisem pola MDMID.

Podanie wartości CINONE jako identyfikatora korelacji ma taki sam efekt, jak *nie* określenie parametru MOCORI, czyli *dowolny* identyfikator korelacji zostanie dopasowany.

Jeśli opcja GMMUC jest określona w parametrze GMO w wywołaniu MQGET, to pole jest ignorowane.

W przypadku powrotu z wywołania MQGET pole MDCID jest ustawione na identyfikator korelacji zwróconego komunikatu (jeśli istnieje).

Mogą być stosowane następujące wartości specjalne:

CINONE

Nie określono identyfikatora korelacji.

Wartość jest binarna zero dla długości pola.

CINEWS

Komunikat jest początkiem nowej sesji.

Ta wartość jest rozpoznawana przez CICS bridge jako wskazującą początek nowej sesji, czyli początek nowej sekwencji komunikatów.

W przypadku wywołania MQGET jest to pole wejściowe/wyjściowe. W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe, jeśli parametr PMNCID *nie* jest określony, oraz pole wyjściowe, jeśli określono wartość PMNCID *jest*. Długość tego pola jest podawana przez LNCID. Wartością początkową tego pola jest CINONE.

MDCSI (10-cyfrowa liczba całkowita ze znakiem)

Ten parametr określa identyfikator zestawu znaków dla danych znakowych w komunikacie.

Uwaga: Dane znakowe w strukturze MQMD i innych strukturach danych produktu IBM MQ, które są parametrami w wywołaniach, muszą znajdować się w zestawie znaków menedżera kolejek. Atrybut ten jest zdefiniowany przez atrybut *CodedCharSetId* menedżera kolejek. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla menedżera kolejek” na stronie 1427](#).

Można użyć następujących wartości specjalnych:

CSQM

Identyfikator zestawu znaków menedżera kolejek.

Dane znakowe w komunikacie znajdują się w zestawie znaków menedżera kolejek.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia tę wartość w deskryptywie MQMD wysłanym z komunikatem na wartość true (prawda) zestawu znaków menedżera kolejek. W wyniku tego wartość CSQM nigdy nie jest zwracana przez wywołanie MQGET.

CINHT

Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w komunikacie znajdują się w tym samym zestawie znaków, co ta struktura. Jest to zestaw znaków menedżera kolejek. (Dotyczy tylko MQMD, CSINHT ma takie samo znaczenie jak CSQM).

Menedżer kolejek zmienia tę wartość w strukturze MQMD wysłanej razem z komunikatem do rzeczywistego identyfikatora zestawu znaków MQMD. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskryptyce MQMD jest *ATBRKR*.

CSEMBD

Identyfikator osadzonego zestawu znaków.

Dane znakowe w komunikacie znajdują się w zestawie znaków z identyfikatorem, który jest zawarty w samych danych komunikatu. W danych komunikatu może znajdować się dowolna liczba identyfikatorów zestawów znaków, które mają zastosowanie do różnych części danych. Ta wartość musi być używana dla komunikatów PCF, które zawierają dane w mieszance zestawów znaków. Komunikaty PCF mają nazwę formatu FMPCF.

Tę wartość należy podać tylko w wywołaniach MQPUT i MQPUT1. Jeśli jest ona określona w wywołaniu MQGET, uniemożliwia ona konwersję komunikatu.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia wartości CSQM i CSINHT w strukturze MQMD wysłanej za pomocą komunikatu zgodnie z wcześniejszym opisem, ale nie powoduje zmiany deskryptora MQMD określonego w wywołaniu MQPUT lub MQPUT1. Żadna inna kontrola nie jest przeprowadzana zgodnie z podaną wartością.

Aplikacje, które pobierają komunikaty, powinny porównać to pole z wartością oczekiwaną przez aplikację; jeśli wartości różnią się, aplikacja może wymagać konwersji danych znakowych w komunikacie.

Jeśli w wywołaniu MQGET określono opcję GMCONV, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest identyfikatorem kodowanego zestawu znaków, do którego dane komunikatu powinny być przekształcane, jeśli jest to konieczne. Jeśli konwersja jest pomyślna lub niepotrzebna, wartość ta pozostaje niezmienną (z tą różnicą, że wartość CSQM lub CSINHT jest przekształcana w wartość rzeczywistą). Jeśli konwersja nie powiedzie się, wartość podana po wywołaniu MQGET reprezentuje identyfikator kodowanego zestawu znaków dla nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W przeciwnym razie jest to pole wyjściowe wywołania MQGET, a także pole wejściowe dla wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest CSQM.

MDENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych komunikatu.

Określa kodowanie liczbowe danych liczbowych w komunikacie; nie ma zastosowania do danych liczbowych w samej strukturze MQMD. Kodowanie numeryczne definiuje reprezentację używaną dla binarnych liczb całkowitych, liczb całkowitych upakowanych liczb całkowitych i zmiennopozycyjnych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Zdefiniowane są następujące wartości specjalne:

ENNAT

Kodowanie rodzimego komputera.

Kodowanie jest domyślne dla języka programowania i komputera, na którym działa aplikacja.

Uwaga: Wartość tej stałej zależy od języka programowania i środowiska. Z tego powodu aplikacje muszą być kompilowane za pomocą plików nagłówkowych, makro, COPY lub INCLUDE odpowiednich dla środowiska, w którym aplikacja będzie uruchamiana.

Aplikacje, które umieszczają komunikaty powinny zwykle określać ENNAT. Aplikacje pobierające komunikaty powinny porównywać to pole z wartością ENNAT; jeśli wartości różnią się, aplikacja może wymagać konwersji danych liczbowych w komunikacie. Opcja GMCONV może być używana do żądania, aby menedżer kolejek przekształcić komunikat w ramach przetwarzania wywołania MQGET.

Jeśli w wywołaniu MQGET określono opcję GMCONV, to pole jest polem wejścia/wyjścia. Wartością określoną przez aplikację jest kodowanie, do którego dane komunikatu powinny zostać przekształcone, jeśli jest to konieczne. Jeśli konwersja jest pomyślna lub niepotrzebna, wartość ta pozostaje niezmienną. Jeśli konwersja nie powiedzie się, wartość podana po wywołaniu MQGET reprezentuje kodowanie nieprzekształconego komunikatu, które jest zwracane do aplikacji.

W innych przypadkach jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest ENNAT.

MDEXP (10-cyfrowa liczba całkowita ze znakiem)

Czas życia komunikatu.

Jest to okres wyrażony w dziesiątych częściach sekundy, ustawiany przez aplikację, która umieszcza komunikat. Komunikat zostaje zakwalifikowany do usunięcia, jeśli nie został usunięty z kolejki docelowej przed upływem tego okresu.

Wartość ta jest zmniejszana, tak aby odzwierciedlała czas, w którym komunikat jest wysyłany w kolejce docelowej, a także w pośrednich kolejkach transmisji, jeśli jest ona do kolejki zdalnej. Może on również zostać zmniejszony przez agenty kanałów komunikatów w celu odzwierciedlenia czasów transmisji, jeśli są one istotne. Podobnie, aplikacja przekazując ten komunikat do innej kolejki może zmniejszyć wartość, jeśli jest to konieczne, o ile zachowała ona komunikat przez istotny czas. Czas utraty ważności jest jednak traktowany jako przybliżony, a wartość nie musi być zmniejszana, aby odzwierciedlać małe przedziały czasu.

Gdy komunikat jest pobierany przez aplikację przy użyciu wywołania MQGET, pole *MDEXP* reprezentuje ilość pierwotnego czasu utraty ważności, który nadal pozostaje.

Po upływie czasu utraty ważności komunikatu, staje się on zakwalifikowany do odrzucenia przez menedżer kolejek. W bieżących implementacjach komunikat jest odrzucany, gdy wystąpi wywołanie funkcji MQGET z przeglądaniem lub nieprzeglądaniem, które zwróciłoby komunikat, gdyby nie utracił on już ważności. Na przykład nieprzeglądanie wywołania MQGET z polem *GMMO* w zestawie MQGMO ustawionym na wartość *MONONE* z kolejki uporządkowanej FIFO spowoduje, że wszystkie wygasłe komunikaty zostaną odrzucone aż do pierwszego nieprzedawnionego komunikatu. W przypadku kolejki uporządkowanej według priorytetu to samo wywołanie odrzuci przedawnione komunikaty o wyższym priorytecie i komunikaty o równym priorytecie, które dotarły do kolejki przed pierwszym nieprzeterminowanym komunikatem.

Komunikat, który utracił ważność, nigdy nie jest zwracany do aplikacji (przy użyciu przeglądania lub wywołania MQGET bez przeglądania), więc wartość w polu *MDEXP* deskryptora komunikatu po pomyślnym wywołaniu MQGET jest większa od zera lub wartość specjalna *EIULIM*.

Jeśli komunikat jest umieszczany w kolejce zdalnej, komunikat może tracić ważność (i być odrzucany), gdy znajduje się on w pośredniej kolejce transmisji, zanim komunikat osiągnie kolejkę docelową.

Raport jest generowany, gdy komunikat, który utracił ważność, jest odrzucany, jeśli w komunikacie określono jedną z opcji raportu ROEXP*. Jeśli żadna z tych opcji nie zostanie określona, taki raport nie zostanie wygenerowany; zakłada się, że komunikat nie będzie już istotny po tym okresie (być może dlatego, że później został zastąpiony przez komunikat).

Każdy inny program, który usuwa komunikaty na podstawie czasu utraty ważności, musi również wysłać odpowiedni komunikat raportu, jeśli zażądano jednego z nich.

Uwaga:

1. Jeśli komunikat jest umieszczany z czasem *MDEXP* o wartości zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny RC2013; w tym przypadku nie jest generowany żaden komunikat raportu.
2. Ponieważ komunikat z czasem utraty ważności, który upłynął, nie może zostać w rzeczywistości usunięty do czasu późniejszego, mogą istnieć komunikaty w kolejce, które przeszły upływ czasu utraty ważności i dlatego nie kwalifikują się do pobrania. Komunikaty te liczą się jednak w stosunku do liczby komunikatów w kolejce dla wszystkich celów, włącznie z wyzwalaniem głębokości.
3. Raport o utracie ważności jest generowany, jeśli zażądano, kiedy komunikat jest rzeczywiście odrzucany, a nie wtedy, gdy staje się on uprawniony do usunięcia.
4. Odrzucanie przedawnionego komunikatu i generowanie raportu o utracie ważności, jeśli zażądano, nigdy nie są częścią jednostki pracy aplikacji, nawet jeśli komunikat został zaplanowany do usunięcia w wyniku wywołania MQGET działającego w ramach jednostki pracy.
5. Jeśli komunikat o prawie ważności jest pobierany za pomocą wywołania MQGET w ramach jednostki pracy, a następnie wycofana jest jednostka pracy, komunikat może zostać zakwalifikowany do usunięcia, zanim będzie można go pobrać ponownie.
6. Jeśli komunikat o prawie ważności jest zablokowany przez wywołanie MQGET z GMLK, komunikat może zostać zakwalifikowany do usunięcia, zanim będzie mógł zostać pobrany przez wywołanie MQGET z kodem GMMUC. Kod przyczyny RC2034 jest zwracany w tym kolejnym wywołaniu MQGET, jeśli tak się stanie.
7. Po pobraniu komunikatu z żądaniem o czasie utraty ważności większym niż zero aplikacja może wykonać jedno z następujących działań, gdy wyśle komunikat odpowiedzi:
 - Skopiuj pozostały czas utraty ważności z komunikatu żądania do komunikatu odpowiedzi.
 - Ustaw czas utraty ważności w komunikacie odpowiedzi na wartość jawną większą niż zero.
 - Ustaw czas utraty ważności w komunikacie odpowiedzi na EIULIM.

Działanie, które ma zostać podjęte, zależy od projektu pakietu aplikacji. Jednak domyślnym działaniem w przypadku umieszczania komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) powinno być zachowanie pozostałego czasu utraty ważności komunikatu i kontynuowanie jego zmniejszenia.

8. Komunikaty wyzwalacza są zawsze generowane z EIULIM.
9. Komunikat (zwykle w kolejce transmisji), który ma nazwę *MDFMT FMXQH*, ma drugi deskryptor komunikatu w *MQXQH*. Z tego powodu są powiązane z nim dwa pola *MDEXP*. W tym przypadku należy odnotować następujące dodatkowe punkty:
 - Gdy aplikacja umieszcza komunikat w kolejce zdalnej, menedżer kolejek umieszcza komunikat początkowo w lokalnej kolejce transmisji, a następnie prefikuje dane komunikatu aplikacji ze strukturą *MQXQH*. Menedżer kolejek ustawia wartości dwóch pól programu *MDEXP* tak, aby były takie same jak wartości określone przez aplikację.
 - Jeśli aplikacja umieszcza komunikat bezpośrednio w lokalnej kolejce transmisji, dane komunikatu muszą już zaczynać się od struktury *MQXQH*, a nazwa formatu musi być typu *FMXQH* (ale menedżer kolejek nie wymusza tego działania). W takim przypadku aplikacja nie musi ustawiać wartości tych dwóch pól *MDEXP*, aby były takie same. (Menedżer kolejek nie sprawdza, czy pole *MDEXP* w tabeli *MQXQH* zawiera poprawną wartość, a nawet czy dane komunikatu są wystarczająco długie, aby można je było dołączyć).
 - Gdy komunikat o nazwie *MDFMT FMXQH* jest pobierany z kolejki (niezależnie od tego, czy jest to kolejka normalna, czy kolejka transmisji), menedżer kolejek zmniejsza *oba* te pola *MDEXP* z czasem spędzonym na oczekiwaniu w kolejce. Jeśli dane komunikatu nie są wystarczająco długie, aby dołączyć pole *MDEXP* w tabeli *MQXQH*, nie jest zgłaszany żaden błąd.
 - Menedżer kolejek używa pola *MDEXP* w oddzielnym deskrypcorze komunikatu (to znaczy nie jest to element w deskrypcorze komunikatu osadzonym w strukturze *MQXQH*) w celu sprawdzenia, czy komunikat jest zakwalifikowany do usunięcia.

- Jeśli początkowe wartości dwóch pól *MDEXP* były różne, to czas *MDEXP* w osobnym deskrypcorze komunikatu jest możliwy, gdy komunikat jest pobierany jako większy od zera (tak więc komunikat nie kwalifikuje się do usunięcia), podczas gdy czas zgodny z polem *MDEXP* w tabeli *MQXQH* upłynął. W tym przypadku pole *MDEXP* w tabeli *MQXQH* jest ustawione na zero.

Rozpoznawana jest następująca wartość specjalna:

EIULIM

Nieograniczony czas życia.

Czas utraty ważności komunikatu jest nieograniczony.

Jest to pole wyjściowe dla wywołania *MQGET* oraz pole wejściowe dla wywołań *MQPUT* i *MQPUT1*. Wartością początkową tego pola jest *EIULIM*.

MDFB (10-cyfrowa liczba całkowita ze znakiem)

Informacja zwrotna lub kod przyczyny.

Ta opcja jest używana z komunikatem typu *MTRPRT* w celu wskazania natury raportu i ma znaczenie tylko w przypadku tego typu komunikatu. Pole może zawierać jedną z wartości *FB** lub jedną z wartości *RC**. Kody opinii są pogrupowane w następujący sposób:

FBNONE

Nie podano informacji zwrotnych.

FBSFST

Najniższa wartość dla informacji zwrotnych generowanych przez system.

FBSLST

Najwyższa wartość dla informacji zwrotnych generowanych przez system.

Zakres kodów sprzężenia zwrotnego generowanych przez system *FBSFST* poprzez *FBSLST* zawiera ogólne kody sprzężenia zwrotnego wymienione w dalszej części tej sekcji (*FB**), a także kody przyczyny (*RC**), które mogą wystąpić, gdy komunikat nie może zostać umieszczony w kolejce docelowej.

FBAFST

Najniższa wartość dla informacji zwrotnych generowanych przez aplikację.

FBALST

Najwyższa wartość dla informacji zwrotnych generowanych przez aplikację.

Aplikacje, które generują komunikaty raportów, nie powinny używać kodów zwrotnych w zakresie systemowym (innym niż *FBQUIT*), chyba że chcą symulować komunikaty raportów wygenerowane przez menedżer kolejek lub agenta kanału komunikatów.

W wywołaniach *MQPUT* lub *MQPUT1* podana wartość musi mieć wartość *FBNONE* lub musi być w zakresie systemowym lub w zakresie aplikacji. Ta opcja jest sprawdzana niezależnie od wartości parametru *MDMT*.

Ogólne kody opinii:

FBCOA

Potwierdzenie przybycia do kolejki docelowej (patrz *ROCOA*).

FBCOD

Potwierdzenie dostarczenia do aplikacji odbierającej (patrz *ROCOD*).

FBEXP

Wiadomość utraciła ważność.

Komunikat został odrzucony, ponieważ nie został usunięty z kolejki docelowej przed upływem czasu jego utraty ważności.

FBPAN

Powiadomienie o działaniu pozytywnym (patrz *ROPAN*).

FBNAN

Powiadomienie o działaniu negatywnym (patrz *RONAN*).

FBQUIT

Aplikacja powinna zakończyć działanie.

Może to być używane przez program do planowania obciążenia w celu kontrolowania liczby działających instancji programu użytkowego. Wystanie komunikatu MTRPRT z tym kodem sprzężenia zwrotnego do instancji programu użytkowego wskazuje na tę instancję, że powinna ona zatrzymać przetwarzanie. Jednak stosowanie tej konwencji jest kwestią dla aplikacji. Nie jest ona wymuszana przez menedżer kolejek.

IMS-kody sprzężenia zwrotnego mostu: Gdy most IMS otrzymuje niezerowy kod rozpoznania IMS-OTMA, most IMS przekształca kod rozpoznania z liczby szesnastkowej na dziesiętną, dodaje wartość FBIERR (300) i umieszcza wynik w polu *MDFB* komunikatu odpowiedzi. Powoduje to, że kod sprzężenia zwrotnego ma wartość z zakresu FBIFST (301) poprzez FBILST (399), gdy wystąpił błąd IMS-OTMA.

Most IMS może generować następujące kody sprzężenia zwrotnego:

FBDLZ

Długość danych wynosi zero.

Długość segmentu była równa zero w danych aplikacji komunikatu.

FBDLN

Ujemna długość danych.

Długość segmentu była ujemna w danych aplikacji komunikatu.

FBDLTB

Zbyt duża długość danych.

Długość segmentu była zbyt duża w danych aplikacji komunikatu.

FBBUFO

Przepełnienie buforu.

Wartość jednego z pól o długości spowodowała przepełnienie buforu komunikatów.

FBLOB1

Długość błędna dla jednego.

Wartość jednego z pól długości była o jeden bajt za krótka.

FBIIH

Struktura MQIIH nie jest poprawna lub nie istnieje.

Pole *MDFMT* w strukturze MQMD określa FMIMS, ale komunikat nie rozpoczyna się od poprawnej struktury MQIIH.

FBNAFI

ID użytkownika nie jest autoryzowany do użycia w produkcie IMS.

Identyfikator użytkownika zawarty w deskrytorze komunikatu MQMD lub hasło zawarte w polu *IIAUT* w strukturze MQIIH nie powiodło się podczas sprawdzania poprawności, które zostało wykonane przez most IMS. W wyniku tego komunikat nie został przekazany do produktu IMS.

FBIERR

Nieoczekiwany błąd zwrócony przez program IMS.

Program IMS zwrócił nieoczekiwany błąd. Zapoznaj się z dziennikiem błędów systemu IBM MQ w systemie, w którym znajduje się most IMS, aby uzyskać więcej informacji na temat tego błędu.

FBIFST

Najniższa wartość dla informacji zwrotnych generowanych przez produkt IMS.

IMS-wygenerowane kody zwrotne zajmują zakres FBIFST (300) poprzez FBILST (399). Kod rozpoznania IMS-OTMA sam w sobie to *MDFB* minus FBIERR.

FBILST

Najwyższa wartość dla informacji zwrotnych generowanych przez produkt IMS.

Kody sprzężenia zwrotnego CICS-most: Następujące kody zwrotne mogą być generowane przez CICS bridge:

FBCAAB

Aplikacja została wstrzymana.

Program użytkowy podany w komunikacie został wstrzymany. Ten kod sprzężenia zwrotnego występuje tylko w polu *DLREA* struktury MQDLH.

FBCANS

Nie można uruchomić aplikacji.

Działanie EXEC CICS LINK dla programu użytkowego określonego w komunikacie nie powiodło się. Ten kod sprzężenia zwrotnego występuje tylko w polu *DLREA* struktury MQDLH.

FBCBRF

Działanie CICS bridge zostało zakończone nieprawidłowo, bez zakończenia normalnego przetwarzania błędów.

FBCCSSE

Niepoprawny identyfikator zestawu znaków.

FBCIHE

Brak struktury nagłówka informacji CICS lub jest ona niepoprawna.

FBCCAE

Długość obszaru commarea CICS jest niepoprawna.

FBCIE

Niepoprawny identyfikator korelacji.

FBCDLQ

Kolejka niewystanych wiadomości nie jest dostępna.

Zadanie CICS bridge nie było w stanie skopiować odpowiedzi na to żądanie do kolejki niedostarczonych komunikatów. Żądanie zostało wycofane.

FBCENE

Kodowanie jest niepoprawne.

FBCINE

Program CICS bridge napotkał nieoczekiwany błąd.

Ten kod sprzężenia zwrotnego występuje tylko w polu *DLREA* struktury MQDLH.

FBCNTA

Nieautoryzowany identyfikator użytkownika lub hasło nie jest poprawne.

Ten kod sprzężenia zwrotnego występuje tylko w polu *DLREA* struktury MQDLH.

BO FBCUBO

Jednostka pracy wycofana.

Kopia zapasowa jednostki pracy została wykonana z jednego z następujących powodów:

- Wykryto awarię podczas przetwarzania innego żądania w ramach tej samej jednostki pracy.
- Abend CICS wystąpił, gdy jednostka pracy była w toku.

FBCUWE

Pole elementu sterującego jednostki pracy *CIUOW* jest niepoprawne.

Kody przyczyny produktu MQ: w przypadku komunikatów o wyjątkach produkt *MDFB* zawiera kod przyczyny MQ. Możliwe są następujące kody przyczyny:

RC2051

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki.

RC2053

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2035

(2035, X'7F3') Brak uprawnień do dostępu.

RC2056

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

RC2048

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

RC2031

(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2030

(2030, X'7EE') Długość komunikatu jest większa niż maksymalna dla kolejki.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest FBNONE.

MDFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych komunikatu.

Jest to nazwa, która może być używana przez nadawcę wiadomości w celu wskazania odbiorcy charakteru danych w komunikacie. Dla nazwy mogą być określone dowolne znaki, które znajdują się w zestawie znaków menedżera kolejek, ale zaleca się, aby nazwa ta była ograniczona do następujących:

- Wielkie litery od A do Z
- Cyfry od 0 do 9

Jeśli używane są inne znaki, może nie być możliwe przetłumaczenie nazwy między zestawami znaków menedżerów kolejek wysyłających i odbierających.

Nazwa powinna być dopełniona odstępami do długości pola lub znakiem o kodzie zero używanym do zakończenia nazwy przed końcem pola; wartość NULL i wszystkie kolejne znaki są traktowane jako znaki puste. Nie należy określać nazwy z odstępami wiodącymi ani odstępami osadzonymi. W przypadku wywołania MQGET menedżer kolejek zwraca nazwę dopełnioną spacjami do długości pola.

Menedżer kolejek nie sprawdza, czy nazwa jest zgodna z zaleceniami opisanymi wcześniej.

Nazwy rozpoczynające się od "MQ" w wielkich, dolnych i mieszanych przypadkach mają znaczenie, które są definiowane przez menedżer kolejek. Nazwy rozpoczynające się od tych liter nie powinny być używane w przypadku własnych formatów. Wbudowane formaty menedżera kolejek to:

FMNONE

Brak nazwy formatu.

Charakter danych nie jest zdefiniowany. Oznacza to, że dane nie mogą być przekształcane, gdy komunikat jest pobierany z kolejki za pomocą opcji GMCONV.

Jeśli parametr GMCONV został określony w wywołaniu MQGET, a zestaw znaków lub kodowanie danych w komunikacie różni się od wartości określonej w parametrze MSGDSC , to komunikat jest zwracany z następującymi kodami zakończenia i przyczyny (przy założeniu, że nie wystąpiły inne błędy):

- Kod zakończenia CCWARN i kod przyczyny RC2110 , jeśli dane FMNONE są na początku komunikatu.
- Kod zakończenia CCOK i kod przyczyny RCNONE, jeśli dane FMNONE są na końcu komunikatu (oznacza to, że jest poprzedzony co najmniej jednym strukturami nagłówka MQ). Struktury nagłówka MQ są przekształcane w żądany zestaw znaków i kodowanie w tym przypadku.

FMADMN

Komunikat żądania/odpowiedzi serwera komend.

Komunikat jest komunikatem żądania lub odpowiedzi serwera komend w formacie programu programowalnego (PCF). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu

MQGET określono opcję GMCONV. Więcej informacji na temat używania programowalnych komunikatów formatu komend zawiera sekcja [Korzystanie z formatów komend programowalnych](#).

FMCIICS

Nagłówek informacji CICS .

Dane komunikatu rozpoczynają się od nagłówka informacyjnego produktu CICS MQCIH, po którym następuje dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *CIFMT* w strukturze MQCIH.

FMCM1

Komunikat odpowiedzi komendy typu 1.

Komunikat to komunikat odpowiedzi serwera komend MQSC, zawierający liczbę obiektów, kod zakończenia i kod przyczyny. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMCM2

Komunikat odpowiedzi komendy typu 2.

Komunikat jest komunikatem odpowiedzi serwera komend MQSC zawierającym informacje na temat żądanych obiektów. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMDLH

Nagłówek niewystanych wiadomości.

Dane komunikatu rozpoczynają się od nagłówka niedostarczonych komunikatów MQDLH. Dane z oryginalnego komunikatu są natychmiast następujące po strukturze MQDLH. Nazwa formatu oryginalnych danych komunikatu jest podawana przez pole *DLFMT* w strukturze MQDLH; szczegółowe informacje na temat tej struktury można znaleźć w sekcji [“MQDLH -nagłówek Dead-letter”](#) na stronie 1077 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

Raporty COA i COD nie są generowane dla komunikatów, które mają *MDFMT* z FMDLH.

FMDH

Nagłówek listy dystrybucyjnej.

Dane komunikatu rozpoczynają się od nagłówka listy dystrybucyjnej MQDH; obejmuje to tablice rekordów MQOR i MQPMR. Po nagłówku listy dystrybucyjnej mogą następować dodatkowe dane. Format dodatkowych danych (jeśli istnieje) jest podany w polu *DHFMT* w strukturze MQDH. Szczegółowe informacje na temat tej struktury zawiera sekcja [“MQDH-nagłówek dystrybucji”](#) na stronie 1072 . Komunikaty o formacie FMDH mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMEVNT

Komunikat zdarzenia.

Komunikat jest komunikatem o zdarzeniu MQ , który raportuje zdarzenie, które wystąpiło. Komunikaty o zdarzeniach mają taką samą strukturę jak komendy programowalne. Więcej informacji na temat tej struktury zawiera sekcja [Struktury komend i odpowiedzi](#). Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Komunikaty zdarzeń Version-1 mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMIMS

Nagłówek informacji IMS .

Dane komunikatu rozpoczynają się od nagłówka informacyjnego IMS MQIIH, po którym następuje dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *IIFMT* w strukturze MQIIH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMIMVS

IMS , zmienna łańcuchowa.

Komunikat jest łańcuchem zmiennej IMS , który jest łańcuchem w postaci 11zzccc, gdzie:

11

jest 2-bajtowym polem długości określaniem łącznej długości elementu łańcucha zmiennej IMS . Ta długość jest równa długości 11 (2 bajty), powiększonej o długość zz (2 bajty), plus długość łańcucha znaków. 11 to dwubajtowa binarna liczba całkowita w kodowaniu określonym w polu *MDENC* .

zz

to dwubajtowe pole zawierające flagi istotne dla IMS. zz jest łańcuchem bajtowym składającym się z dwóch jednobajtowych pól łańcuchowych i jest przesyłany bez zmiany od nadawcy do odbiornika (to znaczy, że zz nie podlega żadnej konwersji).

ccc

to łańcuch znaków o zmiennej długości zawierający znaki 11-4 . ccc znajduje się w zestawie znaków określonym w polu *MDCSI* .

Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMMDE

Rozszerzenie deskryptora komunikatu.

Dane komunikatu rozpoczynają się od rozszerzenia deskryptora komunikatu MQMDE, a następnie są śledzone innymi danymi (zwykle są to dane komunikatu aplikacji). Nazwa formatu, zestaw znaków i kodowanie danych, które są zgodne z MQMDE nadawane są za pomocą pól *MEFMT*, *MECSI* i *MEENC* w MQMDE. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji “MQMDE-Rozszerzenie deskryptora komunikatu” na stronie 1170 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMPCF

Komunikat zdefiniowany przez użytkownika w programowalnym formacie komendy (PCF).

Komunikat jest komunikatem definiowanym przez użytkownika, który jest zgodny ze strukturą komunikatu PCF (programmable command format). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV. Więcej informacji na temat używania programowalnych komunikatów formatu komend zawiera sekcja Korzystanie z formatów komend programowalnych .

FMRMH

Nagłówek komunikatu odwołania.

Dane komunikatu rozpoczynają się od nagłówka komunikatu odwołania MQRMH i są opcjonalnie śledzone przez inne dane. Nazwa formatu, zestaw znaków i kodowanie danych są podawane za pomocą pól *RMFMT*, *RMCSI* i *RMENC* w tabeli MQRMH. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji “MQRMH-nagłówek komunikatu odwołania” na stronie 1218 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMRFH

Reguły i nagłówek formatowania.

Dane komunikatu rozpoczynają się od reguł i nagłówka formatowania MQRFH, a następnie są śledzone innymi danymi. Nazwa formatu, zestaw znaków i kodowanie danych (jeśli istnieją) jest nadawane przez pola *RFFMT*, *RFCSI* i *RFENC* w MQRFH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMRFH2

Reguły i formatowanie nagłówka w wersji 2.

Dane komunikatu rozpoczynają się od reguł version-2 i nagłówka formatowania MQRFH2, a następnie są śledzone innymi danymi. Nazwa formatu, zestaw znaków i kodowanie danych

opcjonalnych (jeśli istnieją) jest nadawane przez pola *RF2FMT*, *RF2CSI* i *RF2ENC* w tabeli MQRFH2. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMSTR

Komunikat składający się całościowo z znaków.

Dane komunikatu aplikacji mogą być łańcuchami SBCS (zestaw znaków jednobajtowych) lub łańcuchami DBCS (zestaw znaków dwubajtowych). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMTM

Komunikat wyzwacza.

Komunikat jest komunikatem wyzwalającym, opisanym przez strukturę MQTM; szczegółowe informacje na temat tej struktury zawiera sekcja “MQTM-komunikat wyzwacza” na stronie 1255 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMWIH

Nagłówek informacji o pracy.

Dane komunikatu rozpoczynają się od nagłówka informacji o pracy MQWIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *WIFMT* w strukturze MQWIH.

FMXQH

Nagłówek kolejki transmisji.

Dane komunikatu rozpoczynają się od nagłówka kolejki transmisji MQXQH. Dane z oryginalnego komunikatu są natychmiast następujące po strukturze MQXQH. Nazwa formatu oryginalnych danych komunikatu jest podawana przez pole *MDFMT* w strukturze MQMD, która jest częścią nagłówka kolejki transmisji MQXQH. Szczegółowe informacje na temat tej struktury można znaleźć w sekcji “MQXQH-nagłówek kolejki transmisji” na stronie 1265 .

Raporty COA i COD nie są generowane dla komunikatów, które mają *MDFMT* z FMXQH.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

MDGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Jest to łańcuch bajtowy używany do identyfikowania określonej grupy komunikatów lub komunikatu logicznego, do którego należy komunikat fizyczny. *MDGID* jest również używany, jeśli dla komunikatu dozwolona jest segmentacja. We wszystkich tych przypadkach wartość *MDGID* ma wartość inną niż NULL, a w polu *MDMFL* ustawiona jest jedna lub więcej następujących opcji:

- MFMIG
- MFLMIG
- MFSEG
- MFLSEG
- MFSEGA

Jeśli żadna z tych opcji nie jest ustawiona, program *MDGID* ma specjalną wartość null GINONE.

To pole nie musi być ustawione przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET parametr MOGRPI *nie* jest określony.

Należy rozważyć użycie tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołania to MQPUT1, aplikacja musi upewnić się, że parametr *MDGID* jest ustawiony na odpowiednią wartość.

Grupy komunikatów i segmenty mogą być przetwarzane poprawnie tylko wtedy, gdy identyfikator grupy jest unikalny. Z tego powodu *aplikacje nie powinny generować własnych identyfikatorów grup* ; zamiast tego aplikacje powinny wykonać jedną z następujących czynności:

- Jeśli określono PMLOGO, menedżer kolejek automatycznie generuje unikalny identyfikator grupy dla pierwszego komunikatu w grupie lub segmencie komunikatu logicznego i używa tego identyfikatora grupy dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego, tak więc aplikacja nie musi podejmować żadnych specjalnych działań. Należy rozważyć zastosowanie tej procedury.
- Jeśli parametr PMLOGO *nie* jest określony, aplikacja powinna zażądać, aby menedżer kolejek wygenerował identyfikator grupy, ustawiając parametr MDGID na GINONE w pierwszym wywołaniu MQPUT lub MQPUT1 w celu uzyskania komunikatu w grupie lub segmencie komunikatu logicznego. Identyfikator grupy zwrócony przez menedżera kolejek na wyjściu z tego wywołania powinien następnie zostać użyty dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego. Jeśli grupa komunikatów zawiera segmentowane komunikaty, to ten sam identyfikator grupy musi być używany dla wszystkich segmentów i komunikatów w grupie.

Jeśli wartość PMLOGO nie jest określona, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności (na przykład w kolejności odwrotnej), ale identyfikator grupy musi być przydzielony przez *pierwsze* wywołanie MQPUT lub MQPUT1, które jest wysyłane dla dowolnego z tych komunikatów.

W przypadku danych wejściowych dla wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w sekcji PMOPT. W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem, jeśli obiekt otwarty jest pojedynczą kolejką, a nie listą dystrybucyjną, ale pozostawia ją niezmienną, jeśli otwarty obiekt jest listą dystrybucyjną. W tym drugim przypadku, jeśli aplikacja musi znać wygenerowane identyfikatory grup, aplikacja musi udostępnić rekordy MQPMR zawierające pole PRGID.

W przypadku danych wejściowych wywołania MQGET menedżer kolejek używa wartości określonej szczegółowo w Tabeli 1. W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Zdefiniowane są następujące wartości specjalne:

GINONE

Nie określono identyfikatora grupy.

Wartość jest binarna zero dla długości pola. Jest to wartość używana dla komunikatów, które nie znajdują się w grupach, nie są segmentami komunikatów logicznych i dla których segmentacja nie jest dozwolona.

Długość tego pola jest nadawana przez LNGID. Wartością początkową tego pola jest GINONE. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDMFL (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatu.

Są to opcje, które określają atrybuty komunikatu lub sterują jego przetwarzaniem. Flagi dzielą się na następujące kategorie:

- Opcja segmentacji
- Flagi statusu

Są one opisane z kolei.

Flagi segmentacji: Gdy komunikat jest zbyt duży dla kolejki, próba umieszczenia komunikatu w kolejce zwykle nie powiedzie się. Segmentacja to technika, za pomocą której menedżer kolejek lub aplikacja dzieli komunikat na mniejsze części zwane segmentami i umieszcza każdy segment w kolejce jako osobny komunikat fizyczny. Aplikacja, która pobiera komunikat, może pobrać segmenty jeden za pomocą jednego lub zażądać menedżera kolejek w celu ponownego złożenia segmentów w jeden komunikat zwracany przez wywołanie MQGET. Ten ostatni jest osiąganym przez określenie opcji GMCMPM w wywołaniu MQGET i dostarczenie buforu, który jest wystarczająco duży, aby pomieścić

pełny komunikat. (Szczegółowe informacje na temat opcji GMCMPM zawiera sekcja “MQGMO-Opcje Get-message” na stronie 1089). Segmentacja komunikatu może wystąpić w wysyłającym menedżerze kolejek, w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek.

Aby sterować segmentacją komunikatu, można określić jedną z następujących opcji:

MFSEGI

Segmentacja jest zablokowana.

Ta opcja uniemożliwia rozbicie komunikatu na segmenty przez menedżer kolejek. Jeśli zostanie określona dla komunikatu, który jest już segmentem, ta opcja uniemożliwia rozbicie segmentu na mniejsze segmenty.

Wartość tej flagi jest binarna zero. Jest to opcja domyślna.

MFSEGA

Segmentacja jest dozwolona.

Ta opcja umożliwia rozbicie komunikatu na segmenty przez menedżer kolejek. Jeśli zostanie określona dla komunikatu, który jest już segmentem, ta opcja umożliwia rozbicie segmentu na mniejsze segmenty. MFSEGA może być ustawiony bez ustawienia MFSEG lub MFLSEG.

Gdy menedżer kolejek segmentuje komunikat, menedżer kolejek włącza flagę MFSEG w kopii deskryptora MQMD, który jest wysyłany z każdym segmentem, ale nie zmienia ustawień tych flag w strukturze MQMD udostępnianej przez aplikację w wywołaniu MQPUT lub MQPUT1 . W przypadku ostatniego segmentu komunikatu logicznego menedżer kolejek włącza również flagę MFLSEG w strukturze MQMD, która jest wysyłana z segmentem.

Uwaga: W przypadku, gdy komunikaty są umieszczane w MFSEGA, ale bez PMLOGO, potrzebne jest staranne działanie. Jeśli komunikat jest następujący:

- Nie jest to segment, oraz
- Nie w grupie, oraz
- Nie są przekazywane,

Aplikacja musi pamiętać o zresetowaniu pola *MDGID* na GINONE przed wywołaniem komendy *each* MQPUT lub MQPUT1 w celu wygenerowania unikalnego identyfikatora grupy, który ma być generowany przez menedżer kolejek dla każdego komunikatu. Jeśli nie jest to zrobione, niepowiązane komunikaty mogą być nieumyślnie zakończone tym samym identyfikatorem grupy, co może prowadzić do nieprawidłowego przetwarzania. Zapoznaj się z opisami pola *MDGID* i opcji PMLOGO, aby uzyskać więcej informacji o tym, kiedy pole *MDGID* musi zostać zresetowane.

Menedżer kolejek rozdziela komunikaty do segmentów w razie potrzeby w celu zapewnienia, że segmenty (oraz wszystkie dane nagłówka, które mogą być wymagane) mieszczą się w kolejce. Istnieje jednak dolna granica wielkości segmentu generowanego przez menedżer kolejek, a tylko ostatni segment utworzony z komunikatu może być mniejszy niż ten limit. (Dolnym limitem dla wielkości segmentu wygenerowanego przez aplikację jest jeden bajt.) Segmenty wygenerowane przez menedżer kolejek mogą mieć nierówną długość. Menedżer kolejek przetwarza komunikat w następujący sposób:

- Formaty zdefiniowane przez użytkownika są podzielone na granice, które są wielokrotnością 16 bajtów. Oznacza to, że menedżer kolejek nie wygeneruje segmentów o wielkości mniejszej niż 16 bajtów (innych niż ostatni segment).
- Wbudowane formaty inne niż FMSTR są dzielone w punktach właściwych dla charakteru prezentowanych danych. Jednak menedżer kolejek nigdy nie splituje komunikatu w środku struktury nagłówka produktu MQ . Oznacza to, że segment zawierający pojedynczą strukturę nagłówka MQ nie może być dalej dzielony przez menedżer kolejek, a w rezultacie minimalny możliwy rozmiar segmentu dla tego komunikatu jest większy niż 16 bajtów.

Drugi lub późniejszy segment wygenerowany przez menedżer kolejek rozpocznie się od jednego z następujących elementów:

- Struktura nagłówka MQ

- Początek danych komunikatu aplikacji
- Część-za pośrednictwem danych komunikatu aplikacji
- Wartość FMSTR jest dzielona bez względu na rodzaj prezentowanych danych (SBCS, DBCS lub mieszane SBCS/DBCS). Jeśli łańcuch jest typu DBCS lub mieszany SBCS/DBCS, może to spowodować, że segmenty, które nie mogą zostać przekształcone z jednego zestawu znaków na inny. Menedżer kolejek nigdy nie splituje komunikatów FMSTR do segmentów o wielkości mniejszej niż 16 bajtów (innych niż ostatni segment).
- Pola *MDFMT*, *MDCSI* i *MDENC* w strukturze MQMD każdego segmentu są ustawiane przez menedżer kolejek w celu poprawnego opisanie danych znajdujących się w *starcie* segmentu. Nazwa formatu będzie nazwą wbudowanego formatu lub nazwą formatu zdefiniowanego przez użytkownika.
- Pole *MDREP* w deskrypcie MQMD segmentów o wartości *MDOFF* większej niż zero jest modyfikowane w następujący sposób:
 - Dla każdego typu raportu, jeśli opcja raportu ma wartość RO* D, ale segment nie może zawierać żadnego z pierwszych 100 bajtów danych użytkownika (czyli danych następujących po strukturach nagłówek MQ, które mogą być obecne), opcja raportu zostanie zmieniona na RO*.

Menedżer kolejek jest zgodny z poprzednio regułami, ale w przeciwnym razie komunikaty nie będą przewijały się w sposób nieprzewidywalny; nie należy zakładać założeń dotyczących miejsca, w którym komunikat jest podzielony.

W przypadku komunikatów *trwałych* menedżer kolejek może wykonywać segmentację tylko w ramach jednostki pracy:

- Jeśli wywołanie MQPUT lub MQPUT1 działa w ramach jednostki pracy zdefiniowanej przez użytkownika, ta jednostka pracy jest używana. Jeśli wywołanie zakończy się niepowodzeniem w ramach procesu segmentacji, menedżer kolejek usuwa wszystkie segmenty, które zostały umieszczone w kolejce w wyniku zakończonego niepowodzeniem wywołania. Jednak niepowodzenie nie uniemożliwia pomyślnego wykonania jednostki pracy.
- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, a nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie zakończy się pomyślnie, menedżer kolejek zatwierdza jednostkę pracy automatycznie (aplikacja nie musi wykonywać tej czynności). Jeśli wywołanie nie powiedzie się, menedżer kolejek wytworzy kopię zapasową jednostki pracy.
- Jeśli wywołanie działa poza zdefiniowaną przez użytkownika jednostką pracy, ale istnieje jednostka pracy zdefiniowana przez użytkownika *nie*, menedżer kolejek nie może wykonać segmentacji. Jeśli komunikat nie wymaga segmentacji, wywołanie może zakończyć się powodzeniem. Jeśli jednak komunikat *nie* wymaga segmentacji, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2255.

W przypadku komunikatów *nietrwałych* menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu przeprowadzenia segmentacji.

Należy zwrócić szczególną uwagę na konwersję danych, które mogą być posegmentowane:

- Jeśli konwersja danych jest wykonywana tylko przez aplikację odbierającą w wywołaniu MQGET, a aplikacja określa opcję GMCMPM, wyjście konwersji danych zostanie przekazane kompletnym komunikatem dla wyjścia do przekształcenia, a fakt, że komunikat był segmentowany, nie będzie widoczny dla wyjścia.
- Jeśli aplikacja odbierająca pobiera jeden segment jednocześnie, to wyjście konwersji danych zostanie wywołane w celu przekształcenia jednego segmentu w danym momencie. W związku z tym wyjście musi być w stanie przekształcić dane w segment niezależnie od danych w żadnym z pozostałych segmentów.

Jeśli charakter danych w komunikacie jest taki, że arbitralna segmentacja danych w granicach 16-bajtowych może spowodować, że segmenty, które nie mogą zostać przekształcone przez wyjście, lub format to FMSTR, a zestaw znaków jest DBCS lub mieszany SBCS/DBCS, to aplikacja

wysyłający powinna utworzyć i umieścić segmenty, określając MFSEGI w celu tłumienia dalszej segmentacji. W ten sposób aplikacja wysyłający może zapewnić, że każdy segment będzie zawierał wystarczającą ilość informacji, aby umożliwić wyjście konwersji danych w celu pomyślnego przekształcenia segmentu.

- Jeśli dla wysyłającego agenta kanału komunikatów (MCA) określono konwersję nadawcy, agent MCA przekształca tylko komunikaty, które nie są segmentami komunikatów logicznych; agent MCA nigdy nie próbuje konwertować komunikatów, które są segmentami.

Ta opcja jest flagą wejściową w wywołaniach MQPUT i MQPUT1 oraz flagą wyjściową wywołania MQGET. W przypadku tego ostatniego wywołania menedżer kolejek również odbija wartość flagi w polu *GMSEG* w produkcie MQGMO.

Wartością początkową tej flagi jest MFSEGI.

flagi statusu: są to flagi, które wskazują, czy komunikat fizyczny należy do grupy komunikatów, czy jest to segment komunikatu logicznego, czy też żaden z nich. W wywołaniu MQPUT lub MQPUT1 lub zwróconej przez wywołanie MQGET można określić co najmniej jedną z następujących wartości:

MFmig

Wiadomość jest członkiem grupy.

MFLmig

Komunikat jest ostatnim komunikatem logicznym w grupie.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza element MFmig w kopii deskryptora MQMD, który jest wysyłany z komunikatem, ale nie zmienia ustawień tych flag w deskrypcji MQMD udostępnionym przez aplikację w wywołaniu MQPUT lub MQPUT1.

Grupa może składać się tylko z jednego komunikatu logicznego. W takim przypadku ustawiona jest wartość MFLmig, ale pole *MDSEQ* ma wartość 1.

MFSEG

Komunikat jest segmentem komunikatu logicznego.

Jeśli wartość MFSEG jest określona bez MFLSEG, długość danych komunikatu aplikacji w segmencie (*wykluczanie* długości wszystkich struktur nagłówek MQ, które mogą być obecne) musi być co najmniej jedna. Jeśli długość wynosi zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny RC2253.

MFLSEG

Komunikat jest ostatnim segmentem komunikatu logicznego.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza element MFSEG w kopii deskryptora MQMD, który jest wysyłany z komunikatem, ale nie zmienia ustawień tych flag w deskrypcji MQMD udostępnionym przez aplikację w wywołaniu MQPUT lub MQPUT1.

Poprawne jest, aby komunikat logiczny składał się tylko z jednego segmentu. W takim przypadku parametr MFLSEG jest ustawiony, ale pole *MDOFF* ma wartość zero.

Jeśli określono MFLSEG, dozwolone jest, aby długość danych komunikatu aplikacji w segmencie (*wykluczona* długości wszystkich struktur nagłówek, które mogą być obecne) była równa zero.

Podczas umieszczania komunikatów aplikacja musi upewnić się, że flagi są poprawnie ustawione. Jeśli określono parametr PMLOGO lub został określony w poprzedzającym wywołaniu MQPUT dla uchwytu kolejki, ustawienia flag muszą być spójne z informacjami o grupach i segmentach zatrzymanych przez menedżer kolejek dla uchwytu kolejki. Następujące warunki mają zastosowanie do *kolejnych* wywołań MQPUT dla uchwytu kolejki, gdy określono PMLOGO:

- Jeśli nie ma żadnej bieżącej grupy lub komunikatu logicznego, wszystkie te opcje (i ich kombinacje) są poprawne.
- Po określeniu MFmig musi ona pozostać w miejscu, dopóki nie zostanie określona wartość MFLmig. Wywołanie nie powiodło się z kodem przyczyny RC2241, jeśli ten warunek nie jest spełniony.
- Po określeniu MFSEG musi on pozostać w miejscu, dopóki nie zostanie określony parametr MFLSEG. Wywołanie nie powiodło się z kodem przyczyny RC2242, jeśli ten warunek nie jest spełniony.

- Po określeniu MFSEG bez MFMIG, MFMIG musi pozostać *off*, dopóki nie zostanie podana wartość MFLSEG. Wywołanie nie powiodło się z kodem przyczyny RC2242, jeśli ten warunek nie jest spełniony.

Tabela 1 przedstawia poprawne kombinacje flag i wartości używane dla różnych pól.

Opcje te są flagami wejściowymi w wywołaniach MQPUT i MQPUT1, a także flagi wyjściowe w wywołaniu MQGET. W tym drugim wywołaniu menedżer kolejek również odbija wartości flag dla pól *GMGST* i *GMSST* w produkcie MQGMO.

Opcje domyślne: można określić, że komunikat ma atrybuty domyślne:

MFBRAK

Brak flag komunikatów (domyślne atrybuty komunikatu).

Powoduje to zahamowanie segmentacji i wskazuje, że komunikat nie znajduje się w grupie i nie jest segmentem komunikatu logicznego. Element MFNONE jest zdefiniowany w celu uzyskania dokumentacji programu pomocy. Nie jest zamierzone, aby ta opcja była używana z innymi, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Pole *MDMFL* jest partycjonowane w podpola; szczegółowe informacje na ten temat zawiera sekcja "[Opcje raportów i flagi komunikatów](#)" na stronie 1462.

Wartością początkową tego pola jest MFNONE. To pole jest ignorowane, jeśli wartość *MDVER* jest mniejsza niż *MDVER2*.

MDMID (24-bajtowy łańcuch bitowy)

Identyfikator komunikatu.

Jest to łańcuch bajtowy używany do odróżniania jednego komunikatu od innego. Ogólnie, dwa komunikaty nie powinny mieć tego samego identyfikatora komunikatu, chociaż nie jest to niedozwolone przez menedżer kolejek. Identyfikator komunikatu jest stałą właściwością komunikatu i utrzymuje się między restartami menedżera kolejek. Ponieważ identyfikator komunikatu jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator komunikatu *nie* jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do drugiego.

W przypadku wywołań MQPUT i MQPUT1, jeśli aplikacja MINONE lub PMNMID jest określona przez aplikację, menedżer kolejek generuje unikalny identyfikator komunikatu.⁴po umieszczeniu komunikatu i umieszcza go w deskrytorze komunikatu wysłanym razem z komunikatem. Menedżer kolejek zwraca również ten identyfikator komunikatu w deskrytorze komunikatu należącym do aplikacji wysyłającej. Aplikacja ta może używać tej wartości do rejestrowania informacji o konkretnych komunikatach, a także do odpowiadania na zapytania z innych części aplikacji.

Jeśli komunikat jest umieszczany w temacie, menedżer kolejek generuje unikalne identyfikatory komunikatów, jeśli jest to konieczne dla każdego opublikowanego komunikatu. Jeśli aplikacja określa identyfikator PMNMID, menedżer kolejek generuje unikalny identyfikator komunikatu, który ma zostać zwrócony przez dane wyjściowe. Jeśli aplikacja MINONE jest określona przez aplikację, wartość pola *MDMID* w strukturze MQMD nie zmienia się po powrocie z wywołania.

Więcej informacji na temat zachowanych publikacji znajduje się w opisie narzędzia PMRET w sekcji [PMOPT](#).

⁴ Produkt *MDMID* wygenerowany przez menedżer kolejek składa się z 4-bajowego identyfikatora produktu (AMQ- lub CSQ- w kodzie ASCII lub EBCDIC, gdzie - reprezentuje pojedynczy pusty znak), po którym następuje implementacja specyficzna dla produktu w postaci unikalnego łańcucha. W produkcie IBM MQ jest to pierwsze 12 znaków nazwy menedżera kolejek oraz wartość pochodząca z zegara systemowego. Dlatego wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się od pierwszych 12 znaków, aby identyfikatory komunikatów były unikalne. Możliwość wygenerowania unikalnego łańcucha zależy również od tego, że zegar systemowy nie jest zmieniany wstecz. Aby wyeliminować możliwość utworzenia identyfikatora komunikatu wygenerowanego przez menedżer kolejek duplikujący wygenerowany przez aplikację, aplikacja powinna unikać generowania identyfikatorów z początkowymi znakami w zakresie od A do I w kodzie ASCII lub EBCDIC (X'41' do X'49' i X'C1' do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, menedżer kolejek generuje unikalne identyfikatory komunikatów w razie potrzeby, ale wartość pola *MDMID* w strukturze MQMD nie zmienia się po powrocie z wywołania, nawet jeśli określono MINONE lub PMNMID. Jeśli aplikacja musi znać identyfikatory komunikatów wygenerowane przez menedżer kolejek, aplikacja musi udostępnić rekordy MQPMR zawierające pole *PRMID*.

Aplikacja wysyłający może również określić konkretną wartość dla identyfikatora komunikatu, innego niż MINONE; spowoduje to zatrzymanie menedżera kolejek generującego unikalny identyfikator komunikatu. Aplikacja, która przekazuje komunikat, może użyć tego narzędzia w celu rozpropagowania identyfikatora komunikatu oryginalnego.

Menedżer kolejek sam nie korzysta z tego pola, z wyjątkiem:

- Generuj unikalną wartość, jeśli jest to wymagane, zgodnie z opisem wcześniej
- Dostarcz wartość do aplikacji, która wydaje żądanie pobrania dla komunikatu.
- Skopiuj wartość do pola *MDCID* dowolnego komunikatu raportu generowanego na temat tego komunikatu (w zależności od opcji *MDREP*).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole *MDMID* w sposób określony przez pole *MDREP* oryginalnego komunikatu, albo RONMI, albo ROPMI. Te aplikacje, które generują komunikaty raportów, również powinny to robić.

W przypadku wywołania MQGET *MDMID* jest jednym z pięciu pól, których można użyć do wybrania konkretnego komunikatu, który ma zostać pobrany z kolejki. Zwykle wywołanie MQGET zwraca następny komunikat w kolejce, ale jeśli wymagany jest konkretny komunikat, można to uzyskać, podając co najmniej jedno z pięciu kryteriów wyboru, w dowolnej kombinacji. Te pola są następujące:

- *MDMID*
- *MDCID*
- *MDGID*
- *MDSEQ*
- *MDOFF*

Aplikacja ustawia jedno lub więcej z tych pól na wymagane wartości, a następnie ustawia odpowiednie opcje dopasowania MO* w polu *GMMO* w produkcie MQGMO w celu wskazania, że te pola powinny być używane jako kryteria wyboru. Tylko komunikaty, które mają określone wartości w tych polach, są kandydatami do pobrania. Wartość domyślna dla pola *GMMO* (jeśli nie została zmieniona przez aplikację) jest zgodna zarówno z identyfikatorem komunikatu, jak i identyfikatorem korelacji.

Zwykle zwracany jest komunikat *pierwszy* w kolejce, który spełnia kryteria wyboru. Jeśli jednak określono parametr GMBRWN, zwracany jest komunikat *next*, który spełnia kryteria wyboru. Skanowanie dla tego komunikatu rozpoczyna się od komunikatu *następującego*, w którym znajduje się bieżąca pozycja kursora.

Uwaga: Kolejka jest skanowana sekwencyjnie dla komunikatu spełniającego kryteria wyboru, dlatego czasy pobierania będą wolniejsze niż w przypadku, gdy nie zostaną określone żadne kryteria wyboru, zwłaszcza jeśli wiele komunikatów ma być skanowanych przed znalezieniem odpowiedniego.

Więcej informacji na temat sposobu użycia kryteriów wyboru w różnych sytuacjach zawiera [Tabela 1](#).

Określenie parametru MINONE jako identyfikatora komunikatu ma taki sam efekt, jak *nie* określenie parametru MOMSGI, czyli *any* identyfikator komunikatu będzie zgodny.

To pole jest ignorowane, jeśli opcja GMMUC jest określona w parametrze *GMO* w wywołaniu MQGET.

W przypadku powrotu z wywołania MQGET pole *MDMID* jest ustawione na identyfikator komunikatu zwróconego przez komunikat (jeśli istnieje).

Można użyć następującej wartości specjalnej:

MINONE

Nie określono identyfikatora komunikatu.

Wartość jest binarna zero dla długości pola.

Jest to pole wejściowe/wyjściowe dla wywołań MQGET, MQPUT i MQPUT1 . Długość tego pola jest podawana przez LNMID. Wartością początkową tego pola jest MINONE.

MDMT (10-cyfrowa liczba całkowita ze znakiem)

Typ wiadomości.

Wskazuje typ komunikatu. Typy komunikatów są pogrupowane w następujący sposób:

MTSFST

Najniższa wartość dla typów komunikatów zdefiniowanych przez system.

MTSLST

Najwyższa wartość dla typów komunikatów zdefiniowanych przez system.

Obecnie w zakresie systemu są zdefiniowane następujące wartości:

MTDGRM

Komunikat nie wymaga odpowiedzi.

Komunikat jest taki, że nie wymaga odpowiedzi.

MTRQST

Komunikat wymagający odpowiedzi.

Komunikat jest taki, że wymaga odpowiedzi.

Nazwa kolejki, do której powinna zostać wysłana odpowiedź, musi być określona w polu *MDRQ* . Pole *MDREP* wskazuje, w jaki sposób mają być ustawione *MDMID* i *MDCID* odpowiedzi.

MTRPLY

Odpowiedź na wcześniejszy komunikat żądania.

Komunikat jest odpowiedzią na wcześniejszy komunikat żądania (MTRQST). Komunikat powinien zostać wysłany do kolejki wskazanej w polu *MDRQ* komunikatu żądania. Pole *MDREP* żądania powinno być używane do sterowania sposobem, w jaki zestaw *MDMID* i *MDCID* odpowiedzi są ustawione.

Uwaga: Menedżer kolejek nie wymusza relacji żądanie-odpowiedź. Jest to odpowiedzialność za aplikację.

MTRPRT

Komunikat raportu.

Komunikat zgłasza oczekiwane lub nieoczekiwane zdarzenie, zwykle związane z jakimś innym komunikatem (na przykład odebrano komunikat żądania, który zawierał niepoprawne dane). Komunikat powinien zostać wysłany do kolejki wskazanej w polu *MDRQ* deskryptora komunikatu oryginalnego komunikatu. Pole *MDFB* powinno być ustawione w taki sposób, aby wskazywało na naturę raportu. Pole *MDREP* oryginalnego komunikatu może być używane do sterowania sposobem, w jaki należy ustawić *MDMID* i *MDCID* komunikatu raportu.

Komunikaty raportów wygenerowane przez menedżera kolejek lub agenta kanału komunikatów są zawsze wysyłane do kolejki produktu *MDRQ* , a pola *MDFB* i *MDCID* są ustawione zgodnie z opisem poprzednio.

Inne wartości w zakresie systemowym mogą być zdefiniowane w przyszłych wersjach interfejsu MQI i są akceptowane przez wywołania MQPUT i MQPUT1 bez błędów.

Możliwe jest również użycie wartości zdefiniowanych przez aplikację. Muszą one znajdować się w następującym zakresie:

MTAFST

Najniższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

MTALST

Najwyższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

W przypadku wywołań MQPUT i MQPUT1 wartość *MDMT* musi należeć albo do zakresu zdefiniowanego przez system, albo do zakresu zdefiniowanego przez aplikację. Jeśli tak nie jest, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2029.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Wartością początkową tego pola jest MTDGRM.

MDOFF (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

Jest to przesunięcie w bajtach danych w komunikacie fizycznym od początku komunikatu logicznego, którego część stanowi część danych. Dane te nazywane są *segmentem*. Przesunięcie mieści się w zakresie od 0 do 999 999 999. Komunikat fizyczny, który nie jest segmentem komunikatu logicznego, ma przesunięcie zerowe.

To pole nie musi być ustawione przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET wartość MOOFFS *nie* jest określona.

Poniżej przedstawiono zalecane sposoby korzystania z tych wywołań w przypadku komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja nie spełnia tych warunków lub wywołanie to MQPUT1, aplikacja musi upewnić się, że parametr *MDOFF* jest ustawiony na odpowiednią wartość.

W przypadku danych wejściowych wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w Tabeli 1. W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem.

W przypadku raportu dotyczącego segmentu komunikatu logicznego pole *MDOLN* (pod warunkiem, że nie jest OLUNDF) jest używane do aktualizowania przesunięcia w informacjach o segmencie zatrzymanych przez menedżer kolejek.

W przypadku danych wejściowych wywołania MQGET menedżer kolejek używa wartości określonej szczegółowo w Tabeli 1. W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Początkowa wartość tego pola wynosi zero. To pole jest ignorowane, jeśli wartość *MDVER* jest mniejsza niż MDVER2.

MDOLN (10-cyfrowa liczba całkowita ze znakiem)

Długość oryginalnego komunikatu.

To pole ma znaczenie tylko w przypadku komunikatów raportu, które są segmentami. Określa długość segmentu wiadomości, do którego odnosi się komunikat raportu; nie określa długości komunikatu logicznego, którego część stanowi segment, ani długość danych w komunikacie raportu.

Uwaga: Podczas generowania komunikatu raportu dla komunikatu, który jest segmentem, menedżer kolejek i agent kanału komunikatów są kopiowane do deskryptora MQMD dla komunikatu raportu *MDGID*, *MDSEQ*, *MDOFF* i *MDMFL*, a pola z pierwotnego komunikatu. W związku z tym komunikat raportu jest również segmentem. Zaleca się, aby aplikacje generujące komunikaty raportów były takie same, a także aby upewnić się, że pole *MDOLN* jest ustawione poprawnie.

Zdefiniowane są następujące wartości specjalne:

OLUNDF

Oryginalna długość komunikatu nie została zdefiniowana.

MDOLN jest polem wejściowym w wywołaniach MQPUT i MQPUT1, ale wartość udostępniana przez aplikację jest akceptowana tylko w określonych okolicznościach:

- Jeśli wstawiany komunikat jest segmentem i jest również komunikatem raportu, menedżer kolejek akceptuje podaną wartość. Wartość musi być następująca:
 - Wartość większa od zera, jeśli segment nie jest ostatnim segmentem
 - Wartość nie mniejsza niż zero, jeśli segment jest ostatnim segmentem.

- Nie mniej niż długość danych znajdujących się w komunikacie

Jeśli te warunki nie są spełnione, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2252.

- Jeśli wysyłany komunikat jest segmentem, ale nie jest komunikatem raportu, menedżer kolejek zignoruje pole i użyje zamiast niej długości danych komunikatu aplikacji.
- We wszystkich innych przypadkach menedżer kolejek ignoruje pole i zamiast niego używa wartości OLUNDF.

To jest pole wyjściowe w wywołaniu MQGET.

Wartością początkową tego pola jest OLUNDF. To pole jest ignorowane, jeśli wartość *MDVER* jest mniejsza niż MDVER2.

MDPAN (28-bajtowy łańcuch znaków)

Nazwa aplikacji umieszczonej w komunikacie.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Format *MDPAN* zależy od wartości *MDPAT*.

Jeśli to pole jest ustawione przez menedżer kolejek (to znaczy dla wszystkich opcji z wyjątkiem PMSETA), jest on ustawiony na wartość określoną przez środowisko:

- W systemie z/OS menedżer kolejek używa:
 - Dla zadania wsadowego z/OS : 8-znakowa nazwa zadania z karty JES JOB.
 - Dla TSO, 7-znakowy identyfikator użytkownika TSO
 - W przypadku systemu CICS: 8-znakowy identyfikator applid, po którym następuje 4-znakowy tranid.
 - W przypadku systemu IMS: 8-znakowy identyfikator systemu IMS , po którym następuje 8-znakowa nazwa PSB.
 - W przypadku XCF: 8-znakowa nazwa grupy XCF, po której następuje 16-znakowa nazwa elementu XCF
 - W przypadku komunikatu wygenerowanego przez menedżer kolejek pierwsze 28 znaków nazwy menedżera kolejek
 - W przypadku rozproszonego kolejkowania bez CICS, 8-znakowa nazwa zadania inicjatora kanału, po której następuje 8-znakowa nazwa modułu umieszczająca w kolejce niedostarczonych komunikatów, po której następuje 8-znakowy identyfikator zadania.
 - W przypadku MQSeries Java -przetwarzanie powiązań języka z IBM MQ for z/OS 8-znakową nazwą zadania przestrzeni adresowej utworzonej dla środowiska UNIX System Services. Zwykle jest to identyfikator użytkownika TSO z dołączonym pojedynczym znakiem liczbowym.

Nazwy lub nazwy są dopełniane spacjami z prawej strony, tak jak to jest w pozostałej części pola. W przypadku, gdy istnieje więcej niż jedna nazwa, nie ma między nimi separatora.

- W systemach operacyjnych DOS i Windows menedżer kolejek używa:
 - W przypadku aplikacji CICS nazwa transakcji CICS
 - W przypadku aplikacji innej niż CICS należy podać co najwyżej 28 znaków pełnej nazwy pliku wykonywalnego.
- W systemie IBM imenedżer kolejek korzysta z pełnej nazwy zadania.
- W systemie HP Integrity NonStop Server menedżer kolejek używa co najwyżej 28 znaków pełnej nazwy pliku wykonywalnego, jeśli jest on dostępny dla menedżera kolejek, a w przeciwnym razie odstęp
- W systemie Systemy UNIX menedżer kolejek używa:
 - W przypadku aplikacji CICS nazwa transakcji CICS

- W przypadku aplikacji innej niż CICS należy podać maksymalnie 14 znaków pełnej nazwy pliku wykonywalnego, jeśli jest ona dostępna dla menedżera kolejek, i odstępy w innych przypadkach (na przykład w systemie AIX).
- W systemie VSE/ESA menedżer kolejek używa 8-znakowego identyfikatora applid, po którym następuje 4-znakowy tranid.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr PMSETA jest określony w parametrze PMO. Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżera kolejek. Jeśli wartość PMSETA nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez LNPN. Początkowa wartość tego pola to 28 znaków odstępu.

MDPAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji, która wstawiła komunikat.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

MDPAT może mieć jeden z następujących typów standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu ATUFST przez ATULST.

równaAIX

AIX (ta sama wartość co ATUNIX).

ATBRKR

Broker.

równaCICS

CICS.

ATCICB

CICS bridge.

ATVSE

CICS/VSE.

ATDOS

Aplikacja IBM MQ MQI client na komputerze PC DOS.

ATDQM

Agent menedżera kolejek rozproszonych.

ATGUAR

Tandem Guardian aplikacja (ta sama wartość jak ATNSK).

równaIMS

Aplikacja IMS.

WIMSB

Most IMS.

ATJAVA

Java.

ATMVS

Aplikacja MVS lub TSO (taka sama wartość jak ATZOS).

NOTATKA

Lotus Notes Aplikacja agenta.

ATNSK

Tandem NonStop, aplikacja jądra.

AT390

Aplikacja OS/390 (taka sama wartość jak ATZOS).

AT400

Aplikacja IBM i .

ATQM

menedżerze kolejek.

równaUNIX

Aplikacja UNIX .

ATVOS

Aplikacja Stratus VOS.

ATWIN

16-bitowa aplikacja Windows .

ATWINT

32-bitowa aplikacja Windows .

ATXCF

XCF.

ATZOS

Aplikacja z/OS .

ATDEF

Domyślny typ aplikacji.

Jest to domyślny typ aplikacji dla platformy, na której działa aplikacja.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

ATUNK

Nieznany typ aplikacji.

Ta wartość może być używana do wskazania, że typ aplikacji jest nieznany, mimo że występują inne informacje o kontekście.

ATUFST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Może wystąpić również następująca wartość specjalna:

ATNCON

W komunikacie nie ma informacji o kontekście.

Ta wartość jest ustawiana przez menedżer kolejek, gdy komunikat jest umieszczany bez kontekstu (oznacza to, że określona jest opcja kontekstu PMNOC).

Po pobraniu komunikatu program *MDPAT* może zostać przetestowany pod kątem tej wartości, aby zdecydować, czy komunikat ma kontekst (zaleca się, aby program *MDPAT* nigdy nie był ustawiony na ATNCON, przez aplikację używała programu PMSETA, jeśli którykolwiek z pozostałych pól kontekstu nie jest pusty).

ATSIB

Wskazuje, że komunikat pochodzi z innego produktu przesyłania komunikatów produktu IBM MQ i został wysłany przez most SIB (Service Integration Bus).

Gdy menedżer kolejek generuje te informacje w wyniku umieszczenia aplikacji, pole jest ustawiane na wartość, która jest określana przez środowisko. Należy zauważyć, że w systemie IBM i jest on ustawiony na wartość AT400; , menedżer kolejek nigdy nie używa ATCICS w systemie IBM i.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr PMSETA jest określony w parametrze *PMO* . Jeśli wartość PMSETA nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDPAT* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDPAT*

przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDPAT*, gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDPAT* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest ustawione na ATNCON.

To jest pole wyjściowe dla wywołania MQGET. Wartością początkową tego pola jest ATNCON.

MDPD (8-bajtowy łańcuch znaków)

Data umieszczenia komunikatu.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Format używany dla daty, w której to pole jest generowane przez menedżer kolejek:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr

rok (cztery cyfry)

MM

miesiąc w roku (01 do 12)

DD

Dzień miesiąca (01 do 31)

Czas Greenwich (GMT) jest używany dla pól *MDPD* i *MDPT*, pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, data jest datą umieszczenia komunikatu, a nie datą, kiedy jednostka pracy została zatwierdzona.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli parametr PMSETA jest określony w parametrze *PMO*. Zawartość pola nie jest sprawdzana przez menedżer kolejek, z tym wyjątkiem, że wszystkie informacje po znaku NULL w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli wartość PMSETA nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDPD*, który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDPD* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDPD*, gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDPD* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez LNPDAT. Początkowa wartość tego pola to 8 znaków odstępu.

MDPER (10-cyfrowa liczba całkowita ze znakiem)

Trwałość komunikatu.

Wskazuje, czy komunikat jest zachowany po awariach systemu i restartach menedżera kolejek. W przypadku wywołań MQPUT i MQPUT1 wartość musi być jedną z następujących wartości:

PEPER

Komunikat jest trwały.

Oznacza to, że komunikat jest zachowany po awariach systemu i restartach menedżera kolejek. Po umieszczeniu komunikacie i zatwierdzeniu jednostki pracy puttera (jeśli komunikat jest umieszczany jako część jednostki pracy), komunikat zostaje zachowany w pamięci dyskowej.

Pozostaje tam do momentu usunięcia komunikatu z kolejki, a także do zatwierdzenia jednostki pracy pobierającej (jeśli komunikat jest pobierany jako część jednostki pracy).

Gdy do kolejki zdalnej wysyłany jest komunikat trwały, do przechowywania komunikatu w każdym menedżerze kolejek wzdłuż trasy do miejsca docelowego jest używany mechanizm przechowywania i przekazywania, aż do momentu, gdy wiadomo, że komunikat dotarł do następnego menedżera kolejek.

Komunikaty trwałe nie mogą być umieszczane w następujących systemach:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, w których poziom struktury narzędzia CF jest mniejszy niż trzy, lub struktura narzędzia CF nie jest odtwarzalna.

Komunikaty trwałe mogą być umieszczane w stałych kolejkach dynamicznych, predefiniowanych kolejkach i kolejkach współużytkowanych, w których poziom struktury narzędzia CF ma wartość 3, a narzędzie CF jest odtwarzalne.

PENPER

Komunikat nie jest trwały.

Oznacza to, że zwykle komunikat nie jest w stanie przetrwać awariach systemu lub restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartu menedżera kolejek zostanie znaleziona nienaruszona kopia komunikatu w pamięci dyskowej.

W specjalnym przypadku kolejek współużytkowanych komunikaty nietrwałe *do* przeżywają restarty menedżerów kolejek w grupie współużytkowania kolejki, ale nie przeżywają niepowodzeń narzędzia CF używanego do przechowywania komunikatów w kolejkach współużytkowanych.

PEQDEF

Komunikat ma trwałość domyślną.

- Jeśli kolejka jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu *DefPersistence* zdefiniowanego w menedżerze kolejek *destination*, który jest właścicielem określonej instancji kolejki, w której umieszczony jest komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu *DefPersistence*, chociaż nie jest to wymagane.

Wartość *DefPersistence* jest kopiowana do pola *MDPER*, gdy komunikat jest umieszczany w kolejce docelowej. Jeśli później zostanie zmieniona opcja *DefPersistence*, komunikaty, które zostały już umieszczone w kolejce, nie będą miały wpływu na te komunikaty.

- Jeśli kolejka nie jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu *DefPersistence* zdefiniowanego w menedżerze kolejek *local*, nawet jeśli docelowy menedżer kolejek jest zdalny.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w definicji *pierwszej* w ścieżce. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Wartość *DefPersistence* jest kopiowana do pola *MDPER* po umieszczeniu w nim komunikacie. Jeśli później zostanie zmieniona wartość *DefPersistence*, komunikaty, które zostały już wprowadzone, nie zostaną naruszone.

Zarówno komunikaty trwałe, jak i nietrwałe mogą istnieć w tej samej kolejce.

Podczas odpowiadania na komunikat aplikacje powinny zwykle używać dla komunikatu odpowiedzi trwałości komunikatu żądania.

W przypadku wywołania MQGET zwracana wartość to PEPER lub PENPER.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest PEQDEF.

MDPRI (10-cyfrowa liczba całkowita ze znakiem)

Priorytet komunikatu.

W przypadku wywołań MQPUT i MQPUT1 wartość ta musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następującej wartości specjalnej:

PRQDEF

Domyślny priorytet kolejki.

- Jeśli kolejka jest kolejką klastra, priorytet komunikatu jest przyjmowany z atrybutu *DefPriority* zdefiniowanego w menedżerze kolejek *destination* , który jest właścicielem określonej instancji kolejki, w której umieszczony jest komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu *DefPriority* , chociaż nie jest to wymagane.

Wartość *DefPriority* jest kopiowana do pola *MDPRI* , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli później zostanie zmieniona opcja *DefPriority* , komunikaty, które zostały już umieszczone w kolejce, nie będą miały wpływu na te komunikaty.

- Jeśli kolejka nie jest kolejką klastra, priorytet komunikatu jest przyjmowany z atrybutu *DefPriority* zgodnie z definicją w *lokalnym* menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, priorytet domyślny jest przyjmowany z wartości tego atrybutu w definicji *pierwszej* w ścieżce. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Wartość *DefPriority* jest kopiowana do pola *MDPRI* po umieszczeniu w nim komunikacie. Jeśli później zostanie zmieniona wartość *DefPriority* , komunikaty, które zostały już wprowadzone, nie zostaną naruszone.

Wartość zwracana przez wywołanie MQGET jest zawsze większa lub równa zero; wartość PRQDEF nigdy nie jest zwracana.

Jeśli komunikat jest umieszczany z priorytetem większym niż maksymalna obsługiwana przez lokalny menedżer kolejek (wartość maksymalna jest podawana przez atrybut menedżera kolejek produktu *MaxPriority*), komunikat jest akceptowany przez menedżer kolejek, ale umieszczany w kolejce w priorytecie maksymalnego priorytetu menedżera kolejek. Wywołanie MQPUT lub MQPUT1 kończy się z kodem CCWARN i kodem przyczyny RC2049. Jednak pole *MDPRI* zachowuje wartość określoną przez aplikację, która wstawiła komunikat.

Podczas odpowiadania na komunikat aplikacje powinny normalnie używać dla komunikatu odpowiedzi priorytet komunikatu żądania. W innych sytuacjach określenie PRQDEF umożliwia strojenie priorytetów bez zmiany aplikacji.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest PRQDEF.

MDPT (8-bajtowy łańcuch znaków)

Czas umieszczenia komunikatu.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Format używany w czasie, gdy pole jest generowane przez menedżer kolejek:

- GGMMSSSTH

gdzie znaki reprezentują (w kolejności):

GG

godzin (od 00 do 23)

MM

minuty (od 00 do 59)

SS

sekundy (od 00 do 59; patrz [uwaga](#))

T

Dziesiątych sekundy (od 0 do 9)

H

Setnych części sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, można w rzadkich przypadkach zwrócić 60 lub 61 sekund w ciągu sekundy w składce *MDPT*. Dzieje się tak wtedy, gdy sekundy przestępne są wstawiane w globalnym standardzie czasu.

Czas Greenwich (GMT) jest używany dla pól *MDPD* i *MDPT* , pod warunkiem, że zegar systemowy jest ustawiony dokładnie na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, godzina jest taka, gdy komunikat został wstawiony, a nie czas, w którym jednostka pracy została zatwierdzona.

W przypadku wywołań *MQPUT* i *MQPUT1* jest to pole wejściowe/wyjściowe, jeśli parametr *PMSETA* jest określony w parametrze *PMO* . Zawartość pola nie jest sprawdzana przez menedżer kolejek, z tym wyjątkiem, że wszystkie informacje po znaku *NULL* w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli wartość *PMSETA* nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko dla danych wyjściowych.

Po pomyślnym zakończeniu wywołania *MQPUT* lub *MQPUT1* w tym polu znajduje się *MDPT* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDPT* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis *PMRET*, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDPT* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDPT* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania *MQGET*. Długość tego pola jest podana przez *LNPTIM*. Początkowa wartość tego pola to 8 znaków odstępu.

MDREP (10-cyfrowa liczba całkowita ze znakiem)

Opcje dla komunikatów raportu.

Komunikat raportu to komunikat o innym komunikacie, który jest używany do informowania aplikacji o oczekiwanych lub nieoczekiwanych zdarzeniach, które odnoszą się do oryginalnego komunikatu. Pole *MDREP* umożliwia aplikacji wysyłanie oryginalnego komunikatu w celu określenia, które komunikaty raportów są wymagane, czy dane komunikatu aplikacji mają być zawarte w nich, a także (dla obu raportów i odpowiedzi), w jaki sposób mają być ustawione identyfikatory komunikatów i korelacji w komunikacie raportu lub odpowiedzi. Można zażądać dowolnego lub wszystkiego (lub żadnego) z następujących typów komunikatów raportu:

- Wyjątek
- Termin ważności
- Potwierdź po przybyciu (COA)
- Potwierdzenie dostarczenia (COD)
- Powiadomienie o działaniu pozytywnym (PAN)
- Powiadomienie o działaniu negatywnym (NAN)

Jeśli wymagane jest więcej niż jeden typ komunikatu raportu lub wymagane są inne opcje raportu, wartości można dodawać razem (nie należy dodawać tej samej stałej więcej niż raz).

Aplikacja, która odbiera komunikat raportu, może określić przyczynę wygenerowania raportu, sprawdzając pole *MDFB* w strukturze MQMD. Więcej informacji na ten temat zawiera pole *MDFB*.

Użycie opcji raportu podczas umieszczania komunikatu w temacie może spowodować wygenerowanie i wysłanie do aplikacji jednego lub wielu komunikatów raportu. Wynika to z faktu, że komunikat publikacji może być wysyłany do jednego lub wielu aplikacji subskrybujących.

Opcje wyjątku: Można określić jedną z następujących opcji, aby zażądać komunikatu o wyjątku.

ROACTIVITY

Wymagane raporty aktywności

Ta opcja raportu umożliwia wygenerowanie raportu aktywności, gdy komunikat z tym zestawem opcji raportu jest przetwarzany przez aplikacje pomocnicze.

Komunikaty z tym zestawem opcji raportu muszą być akceptowane przez dowolny menedżer kolejek, nawet jeśli nie "rozumie" tej opcji. Pozwala to na ustawienie opcji raportu dla dowolnego komunikatu użytkownika, nawet jeśli są one przetwarzane przez poprzednie menedżery kolejek. Aby to osiągnąć, opcja raportu jest umieszczana w podpolu ROAUM.

Jeśli proces (menedżer kolejek lub proces użytkownika) wykonuje działanie na komunikacie z zestawem ROACT, może zdecydować o wygenerowaniu i umieszczeniu raportu aktywności.

Opcja raportu aktywności umożliwia śledzenie trasy każdego komunikatu w sieci menedżera kolejek. Opcja raportu może być określona dla dowolnego bieżącego komunikatu użytkownika i od razu mogą one zacząć obliczać trasę komunikatu przez sieć. Jeśli aplikacja generująca komunikat nie może przełączać się na raporty aktywności, może być ona włączona przy użyciu wyjścia funkcji API, które jest dostarczane przez administratorów menedżera kolejek.

Do raportów dotyczących działań stosuje się kilka warunków:

1. Trasa będzie mniej szczegółowa, jeśli w sieci jest mniej menedżerów kolejek, które są w stanie generować raporty aktywności.
2. Raporty dotyczące działań mogą nie być łatwo zamawiane w celu określenia podjętej trasy.
3. Raporty dotyczące działań mogą nie być w stanie znaleźć trasy dożądanego miejsca docelowego.

ROEXC

Wymagane raporty o wyjątkach.

Ten typ raportu może być generowany przez agenta kanału komunikatów, gdy komunikat jest wysyłany do innego menedżera kolejek, a komunikat nie może zostać dostarczony do określonej kolejki docelowej. Na przykład kolejka docelowa lub pośrednia kolejka transmisji może być pełna, albo komunikat może być zbyt duży dla kolejki.

Generowanie komunikatu raportu o wyjątkach zależy od trwałości oryginalnego komunikatu oraz szybkości kanału komunikatów (normalnego lub szybkiego), za pośrednictwem którego oryginalny komunikat jest przemieszczany:

- W przypadku wszystkich komunikatów trwałych oraz w przypadku komunikatów nietrwałych podróżujących za pośrednictwem zwykłych kanałów komunikatów raport o wyjątku jest generowany *tylko*, jeśli działanie określone przez aplikację wysyłającą dla warunku błędu może zostać pomyślnie zakończone. Aplikacja wysyłająca może określić jedno z następujących działań, aby sterować rozporządzaniem oryginalnego komunikatu w przypadku wystąpienia warunku błędu:
 - RODLQ (powoduje to, że oryginalny komunikat jest umieszczany w kolejce niedostarczonych komunikatów).
 - RODISC (powoduje, że oryginalny komunikat jest usuwany).

Jeśli działanie określone przez aplikację wysyłającą nie może zostać zakończone pomyślnie, oryginalny komunikat jest pozostawiony w kolejce transmisji i nie zostanie wygenerowany żaden komunikat o raporcie o wyjątku.

- W przypadku komunikatów nietrwałych podróżujących za pomocą szybkich kanałów komunikatów oryginalny komunikat jest usuwany z kolejki transmisji, a raport wyjątku wygenerowany *nawet wtedy, gdy* nie można pomyślnie zakończyć określonego działania dla warunku błędu. Na przykład, jeśli określono parametr RODLQ, ale nie można umieścić oryginalnego komunikatu w kolejce niedostarczonych komunikatów, ponieważ (powiedzmy) kolejka jest pełna, generowany jest komunikat o wyjątku, a oryginalny komunikat został usunięty.

Więcej informacji na temat normalnych i szybkich kanałów komunikatów zawiera sekcja [Trwałość komunikatów](#).

Raport o wyjątku nie jest generowany, jeśli aplikacja, która umieściła oryginalny komunikat, może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1.

Aplikacje mogą również wysyłać raporty o wyjątkach, aby wskazać, że odebrany komunikat nie może zostać przetworzony (na przykład dlatego, że jest to transakcja debetowa, która powodowałaby przekroczenie limitu kredytowego konta).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie określaj więcej niż jeden z wierszy ROEXC, ROEXCD i ROEXCF.

ROEXCD

Raporty o wyjątkach z danymi wymaganymi.

Jest to takie samo, jak ROEXC, z tym że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie określaj więcej niż jeden z wierszy ROEXC, ROEXCD i ROEXCF.

ROEXCF

Raporty o wyjątkach z pełnymi danymi są wymagane.

Jest to takie samo, jak ROEXC, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie określaj więcej niż jeden z wierszy ROEXC, ROEXCD i ROEXCF.

Opcje utraty ważności: Można określić jedną z następujących opcji, aby zażądać komunikatu o utracie ważności.

ROEXP

Wymagane raporty o utracie ważności.

Ten typ raportu jest generowany przez menedżer kolejek, jeśli komunikat został odrzucony przed dostarczeniu do aplikacji, ponieważ upłynął jego czas utraty ważności (patrz pole *MDEXP*). Jeśli ta opcja nie zostanie ustawiona, komunikat raportu nie zostanie wygenerowany, jeśli z tego powodu zostanie usunięty komunikat (nawet jeśli określono jedną z opcji ROEXC*).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy określać więcej niż jednego z następujących elementów: ROEXP, ROEXPD i ROEXPF.

ROEXPD

Raporty o utracie ważności z danymi wymaganymi.

Jest to takie samo, jak ROEXP, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie należy określać więcej niż jednego z następujących elementów: ROEXP, ROEXPD i ROEXPF.

ROEXPF

Raporty o utracie ważności z pełnymi danymi są wymagane.

Jest to takie samo, jak ROEXP, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie należy określać więcej niż jednego z następujących elementów: ROEXP, ROEXPD i ROEXPF.

Opcje potwierdzania po przybyciu: Można określić jedną z następujących opcji, aby zażądać komunikatu potwierdzenia odbioru w dniu przyjazdu.

ROCOA

Wymagane są raporty z potwierdzeniem odbioru.

Ten typ raportu jest generowany przez menedżer kolejek, który jest właścicielem kolejki docelowej, gdy komunikat jest umieszczany w kolejce docelowej. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest umieszczany jako część jednostki pracy, a kolejka docelowa jest kolejką lokalną, komunikat raportu COA wygenerowany przez menedżer kolejek staje się dostępny do pobrania tylko wtedy, gdy i kiedy jednostka pracy jest zatwierdzana.

Raport COA nie jest generowany, jeśli pole *MDFMT* w deskryptorze komunikatu to FMXQH lub FMDLH. Zapobiega to generowaniu raportu COA, jeśli komunikat jest umieszczany w kolejce transmisji, lub jest niedostarczalny i umieszczony w kolejce niedostarczonych komunikatów.

Nie określaj więcej niż jednego z ROCOA, ROCOAD i ROCOAF.

ROCOAD

Potwierdzanie raportów w momencie przybycia z wymaganymi danymi.

Jest to ta sama wartość co ROCOA, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Nie określaj więcej niż jednego z ROCOA, ROCOAD i ROCOAF.

ROCOAF

Potwierdzanie raportów w momencie przybycia z pełnymi danymi wymaganymi.

Jest to takie samo, jak ROCOA, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Nie określaj więcej niż jednego z ROCOA, ROCOAD i ROCOAF.

Odrzuć i opcje utraty ważności: Można określić następującą opcję, aby ustawić flagę czasu utraty ważności i odrzuć dla komunikatów raportu.

ROPDAE

Ustaw flagę czasu utraty ważności komunikatu raportu i odrzuć flagę.

Ta opcja zapewnia, że komunikaty i komunikaty odpowiedzi dziedziczą czas utraty ważności i odrzuć flagę (odrzucaj lub nie), z ich oryginalnych komunikatów. Za pomocą tego zestawu opcji można ustawić, raportować i odpowiadać na komunikaty:

1. Dziedzicz flagę RODISC (jeśli została ustawiona).
2. Dziedzicz pozostały czas utraty ważności komunikatu, jeśli komunikat nie jest raportem dotyczącym utraty ważności. Jeśli komunikat jest raportem dotyczącym utraty ważności, czas utraty ważności jest ustawiony na 60 sekund.

Z tym zestawem opcji stosuje się następujące elementy:

Uwaga:

1. Komunikaty raportów i odpowiedzi są generowane z flagą odrzucenia i wartością utraty ważności i nie mogą pozostawać w systemie.
2. Komunikaty śledzenia trasy są uniemożliwiane przez dotarcie do kolejek docelowych na menedżerach kolejek z włączoną obsługą trasy niesledzenia.
3. Nie można zapełniać kolejek za pomocą raportów, których nie można dostarczyć, jeśli połączenia komunikacyjne są zerwane.
4. Odpowiedzi serwera komend dziedziczą pozostały okres ważności żądania.

Opcje potwierdzania dostarczenia: Można określić jedną z następujących opcji, aby zażądać komunikatu potwierdzania dotyczącego dostarczenia.

ROCOD

Wymagane są raporty z potwierdzeniem dostarczenia.

Ten typ raportu jest generowany przez menedżer kolejek, gdy aplikacja pobiera komunikat z kolejki docelowej w sposób, który powoduje usunięcie komunikatu z kolejki. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest pobierany jako część jednostki pracy, komunikat raportu jest generowany w ramach tej samej jednostki pracy, tak aby raport nie był dostępny do momentu zatwierdzenia jednostki pracy. Jeśli jednostka pracy jest wycofana, raport nie jest wysyłany.

Raport COD nie jest generowany, jeśli pole *MDFMT* w deskrypcji komunikatu ma wartość *FMDLH*. Zapobiega to generowaniu raportu COD, jeśli komunikat jest niedostarczalny i jest umieszczany w kolejce niedostarczonych komunikatów.

ROCOD nie jest poprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jeden ROCOD, ROCODD i ROCODF więcej niż jeden.

ROCODD

Potwierdzanie raportów w czasie dostarczania z wymaganymi danymi.

Jest to ta sama wartość, co ROCOD, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest zawarte w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka produktu MQ, są one dołączane do komunikatu raportu, a także do 100 bajtów danych aplikacji.

Jeśli w wywołaniu MQGET dla oryginalnego komunikatu określono parametr *GMATM*, a wczytany komunikat jest obcięty, to ilość danych komunikatu aplikacji umieszczanych w komunikacie raportu jest minimalna:

- Długość oryginalnego komunikatu
- 100 bajtów.

Parametr ROCODD nie jest poprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jeden ROCOD, ROCODD i ROCODF więcej niż jeden.

ROCODF

Potwierdzanie raportów na temat dostarczania z pełnymi danymi wymaganymi.

Jest to ta sama wartość, co ROCOD, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są zawarte w komunikacie raportu.

Parametr ROCODF nie jest poprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie określaj więcej niż jeden ROCOD, ROCODD i ROCODF więcej niż jeden.

Opcje powiadamiania o działaniu: Można określić jedną lub obie z poniższych opcji, aby zażądać, aby aplikacja odbierająca wysłała komunikat o pozytywnym działaniu lub komunikat o negatywnym działaniu.

ROPAN

Wymagane są raporty dotyczące działań pozytywnych.

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie zostało wykonane pomyślnie. Aplikacja generujący raport określa, czy dane mają zostać dołączone do raportu.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby odpowiedzialność za pobranie raportu jest generowana przez aplikację pobierający.

RONAN

Wymagane są raporty o działaniu negatywnym.

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie *nie* zostało wykonane pomyślnie. Aplikacja generujący raport określa, czy dane mają zostać dołączone do raportu. Na przykład może być požądane dołączenie niektórych danych wskazujących, dlaczego żądanie nie mogło zostać wykonane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby odpowiedzialność za pobranie raportu jest generowana przez aplikację pobierający.

Ustalenie, jakie warunki odpowiadają działaniu pozytywnym i które odpowiadają negatywnym działaniom, jest obowiązkiem zgłoszenia. Jednakże zaleca się, aby w przypadku gdy wniosek został przeprowadzony tylko częściowo, w razie potrzeby należy wygenerować raport NAN, a nie raport PAN. Zaleca się również, aby każdy możliwy warunek był zgodny z działaniem pozytywnym, albo działaniem negatywnym, ale nie obu jednocześnie.

Opcje identyfikatora komunikatu: Można określić jedną z następujących opcji, aby określić, w jaki sposób *MDMID* ma być ustawiony komunikat raportu (lub komunikat odpowiedzi).

RONMI

Identyfikator nowego komunikatu.

Jest to działanie domyślne, które wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, zostanie wygenerowany nowy *MDMID* dla komunikatu lub komunikatu odpowiedzi.

ROPMI

Przekaz identyfikator komunikatu.

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, *MDMID* tego komunikatu ma zostać skopiowany do *MDMID* komunikatu raportu lub odpowiedzi.

MsgId komunikatu publikacji będzie się różnić dla każdego subskrybenta, który otrzymuje kopię publikacji, dlatego *MsgId* skopiowana do raportu lub komunikat odpowiedzi będzie się różnić dla każdego z tych publikacji.

Jeśli ta opcja nie zostanie podana, przyjmowana jest wartość RONMI.

Opcje identyfikatora korelacji: Można określić jedną z następujących opcji, aby określić, w jaki sposób *MDCID* ma być ustawiony komunikat raportu (lub komunikat odpowiedzi).

ROCMTC

Kopiuje identyfikator komunikatu do identyfikatora korelacji.

Jest to działanie domyślne i wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, to *MDMID* tego komunikatu ma zostać skopiowany do *MDCID* komunikatu raportu lub odpowiedzi.

MsgId komunikatu publikacji będzie się różnić dla każdego subskrybenta, który otrzymuje kopię publikacji, dlatego *MsgId* skopiowana do *CorrelId* komunikatu raportu lub odpowiedzi będzie się różnić dla każdego z nich.

ROPCI

Przekaz identyfikator korelacji.

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, *MDCID* tego komunikatu ma zostać skopiowany do *MDCID* komunikatu raportu lub odpowiedzi.

MDCID komunikatu publikacji będzie specyficzny dla subskrybenta, o ile nie używa opcji *SOSCID* i ustawia pole *SCDIC* w *MQSD* na *CINONE*. Z tego powodu możliwe jest, że *MDCID* skopiowana do *MDCID* komunikatu raportu lub odpowiedzi będzie inna dla każdego z nich.

Jeśli ta opcja nie zostanie podana, przyjmowana jest wartość *ROCMTC*.

Zaleca się, aby serwery odpowiadały żądaniom lub wygenerowały komunikaty raportów w celu sprawdzenia, czy opcje *ROPMI* lub *ROPCI* zostały ustawione w oryginalnym komunikacie. Jeśli były, serwery powinny podjąć działania opisane dla tych opcji. Jeśli żadna z tych wartości nie zostanie ustawiona, serwery powinny podjąć odpowiednie działanie domyślne.

: Można określić jedną z następujących opcji, aby sterować rozporządzaniem oryginalnego komunikatu, gdy nie można go dostarczyć do kolejki docelowej. Te opcje mają zastosowanie tylko do sytuacji, które spowodowałyby wygenerowanie komunikatu o wyjątku, jeśli żądanie zostało wystane przez aplikację wysyłającą. Aplikacja może ustawić opcje rozporządzania niezależnie od żądania raportów o wyjątkach.

RODLQ

Umieść komunikat w kolejce niedostarczonych komunikatów.

Jest to działanie domyślne, które wskazuje, że komunikat powinien zostać umieszczony w kolejce niedostarczonych komunikatów, jeśli komunikat nie może zostać dostarczony do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieszcza oryginalny komunikat, nie może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie *MQPUT* lub *MQPUT1*. Generowany jest komunikat o wyjątku, o ile został on zażądany przez nadawcę.
- Gdy aplikacja, która umieszcza oryginalny komunikat, została wstawiona do tematu

Wygenerowany zostanie komunikat o wyjątku, o ile został on zażądany przez nadawcę.

RODISC

Odrzuć komunikat.

Oznacza to, że komunikat powinien zostać usunięty, jeśli nie może zostać dostarczony do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieszcza oryginalny komunikat, nie może być powiadamiana synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie *MQPUT* lub *MQPUT1*. Generowany jest komunikat o wyjątku, o ile został on zażądany przez nadawcę.
- Gdy aplikacja, która umieszcza oryginalny komunikat, została wstawiona do tematu

Wygenerowany zostanie komunikat o wyjątku, o ile został on zażądany przez nadawcę.

Jeśli pożądane jest zwrócenie oryginalnego komunikatu do nadawcy, bez umieszczania oryginalnego komunikatu w kolejce niedostarczonych komunikatów, nadawca powinien określić *RODISC* z *ROEXCF*.

Opcja domyślna: można określić następujące opcje, jeśli nie są wymagane żadne opcje raportu:

RONONE

Nie jest wymagane żadne raporty.

Ta wartość może być używana do wskazania, że nie określono żadnych innych opcji. *RONONE* jest zdefiniowany w dokumentacji programu pomocy. Ta opcja nie jest przeznaczona do użycia z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Informacje ogólne:

1. Wszystkie wymagane typy raportów muszą być specjalnie wymagane przez aplikację wysyłającą oryginalny komunikat. Na przykład, jeśli zażądano raportu *COA*, ale raport o wyjątku nie jest, raport *COA* jest generowany, gdy komunikat jest umieszczany w kolejce docelowej, ale nie jest generowany żaden raport o wyjątku, jeśli kolejka docelowa jest zapelniona po przybyciu komunikatu do kolejki docelowej. Jeśli nie ustawiono opcji *MDREP*, menedżer kolejek lub agent kanału komunikatów (*MCA*) nie wygeneruje komunikatów raportu.

Niektóre opcje raportu można określić nawet wtedy, gdy lokalny menedżer kolejek nie rozpoznaje ich. Jest to przydatne, gdy opcja ma być przetwarzana przez menedżer kolejek *docelowy*. Więcej szczegółów na ten temat zawiera sekcja “Opcje raportów i flagi komunikatów” na stronie 1462.

Jeśli zażądano komunikatu raportu, w polu *MDRQ* należy podać nazwę kolejki, do której powinien zostać wysłany raport. Po odebraniu komunikatu z raportem rodzaj raportu można określić, badając pole *MDFB* w deskrypcji komunikatu.

2. Jeśli menedżer kolejek lub agent MCA generujący komunikat raportu nie może umieścić komunikatu raportu w kolejce odpowiedzi (na przykład dlatego, że kolejka odpowiedzi lub kolejka transmisji jest pełna), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów. Jeśli *również* nie powiedzie się, lub nie ma kolejki niedostarczonych komunikatów, działanie jest zależne od typu komunikatu raportu:
 - Jeśli komunikat raportu jest raportem o wyjątku, komunikat, który spowodował wygenerowanie raportu o wyjątkach, jest pozostawiony w kolejce transmisji, co zapewnia, że komunikat nie zostanie utracony.
 - Dla wszystkich pozostałych typów raportów komunikat raportu jest odrzucany, a przetwarzanie jest kontynuowane normalnie. Dzieje się tak dlatego, że oryginalny komunikat został już bezpiecznie dostarczony (dla komunikatów raportu COA lub COD) lub nie jest już zainteresowany (dla komunikatu o utracie ważności).

Gdy komunikat raportu zostanie pomyślnie umieszczony w kolejce (kolejka docelowa lub pośrednia kolejka transmisji), komunikat nie będzie już podlegał specjalnym przetwarzaniu; jest traktowany tak samo jak każdy inny komunikat.

3. Po wygenerowaniu raportu kolejka *MDRQ* jest otwierana, a komunikat raportu jest umieszczany przy użyciu uprawnień *MDUID* w strukturze MQMD komunikatu, co powoduje, że raport jest generowany, z wyjątkiem następujących przypadków:
 - Raporty o wyjątkach wygenerowane przez odbierający agent MCA są umieszczane z dowolnymi uprawnieniami używanego przez agenta MCA podczas próby umieszczenia komunikatu powodującego raport. Atrybut kanału *CDPA* określa używany identyfikator użytkownika.
 - Raporty COA wygenerowane przez menedżera kolejek są umieszczane z dowolnymi uprawnieniami, gdy komunikat powodujący wygenerowanie raportu został wygenerowany przez menedżer kolejek generujący raport. Na przykład, jeśli komunikat został umieszczony przez odbierającego agenta MCA przy użyciu identyfikatora użytkownika MCA, menedżer kolejek umieszcza raport COA przy użyciu identyfikatora użytkownika MCA.

Aplikacje generujące raporty powinny normalnie korzystać z tego samego uprawnienia, co do wygenerowania odpowiedzi. Zwykle jest to uprawnienie identyfikatora użytkownika w oryginalnym komunikacie.

Jeśli raport musi podróżować do miejsca docelowego, nadawcy i odbiorcy mogą zdecydować, czy go zaakceptować, w taki sam sposób, jak w przypadku innych komunikatów.

4. Jeśli zażądano komunikatu raportu z danymi:
 - Komunikat raportu jest zawsze generowany wraz z ilością danych żądanych przez nadawcę oryginalnego komunikatu. Jeśli komunikat raportu jest zbyt duży dla kolejki odpowiedzi, przetwarzanie opisane wcześniej; komunikat raportu nigdy nie jest obcinany, aby można było zmieścić się w kolejce odpowiedzi.
 - Jeśli *MDFMT* oryginalnego komunikatu to FMXQH, dane zawarte w raporcie nie zawierają tabeli MQXQH. Dane raportu rozpoczynają się od pierwszego bajtu danych poza MQXQH w oryginalnym komunikacie. Dzieje się tak, czy kolejka jest kolejką transmisji.
5. Jeśli w kolejce odpowiedzi zostanie odebrany komunikat COA, COD lub raportu o utracie ważności, to zostanie zagwarantowane, że oryginalny komunikat został dostarczony, został dostarczony lub utracił ważność, w zależności od przypadku. Jeśli jednak zażądano co najmniej jednego z tych komunikatów raportu, a *nie* zostanie odebrany, nie można założyć odwrotnej sytuacji, ponieważ mogło wystąpić jedno z następujących zdarzeń:
 - a. Komunikat raportu jest wstrzymany, ponieważ odsyłacz jest wyłączony.

- b. Komunikat raportu jest wstrzymany, ponieważ warunek blokowania istnieje w pośredniej kolejce transmisji lub w kolejce odpowiedzi (na przykład kolejka jest zapełniona lub zablokowana dla operacji put).
- c. Komunikat raportu znajduje się w kolejce niedostarczonych komunikatów.
- d. Gdy menedżer kolejek próbował wygenerować komunikat raportu, nie mógł go umieścić w odpowiedniej kolejce i nie był w stanie umieścić go w kolejce niedostarczonych komunikatów, dlatego nie można było wygenerować komunikatu raportu.
- e. Wystąpił błąd menedżera kolejek między raportowaniem działania (nadejściem, dostawą lub utratą ważności) a wygenerowaniem odpowiedniego komunikatu raportu. (Nie zdarza się to dla komunikatów raportu COD, jeśli aplikacja wczytuje oryginalny komunikat w jednostce pracy, ponieważ komunikat raportu COD jest generowany w ramach tej samej jednostki pracy).

Komunikaty o wyjątkach mogą być przechowywane w ten sam sposób z powodów 1, 2 i 3 wcześniej. Jeśli jednak agent MCA nie może wygenerować komunikatu raportu o wyjątkach (komunikat raportu nie może zostać umieszczony w kolejce odpowiedzi lub w kolejce niedostarczonych komunikatów), oryginalny komunikat pozostaje w kolejce transmisji u nadawcy, a kanał jest zamknięty. Dzieje się tak niezależnie od tego, czy komunikat raportu miał być generowany podczas wysyłania, czy też odbierającego końca kanału.

- 6. Jeśli oryginalny komunikat jest tymczasowo zablokowany (w wyniku czego generowany jest komunikat o wyjątku, a oryginalny komunikat jest umieszczany w kolejce niedostarczonych komunikatów), ale blokada i aplikacja odczyta oryginalny komunikat z kolejki niedostarczonych komunikatów i umieszcza go ponownie w miejscu docelowym, mogą wystąpić następujące działania:
 - Mimo że wygenerowano komunikat o wyjątku, oryginalny komunikat w końcu dotarł do miejsca docelowego.
 - W odniesieniu do pojedynczego oryginalnego komunikatu generowany jest więcej niż jeden komunikat raportu o wyjątku, ponieważ oryginalny komunikat może napotkać inną blokadę później.

Zgłaszanie komunikatów podczas wprowadzania do tematu:

- 1. Raporty mogą być generowane podczas umieszczania komunikatu w temacie. Ten komunikat zostanie wysłany do wszystkich subskrybentów tematu, który może być równy zero, jeden lub wiele. Należy to wziąć pod uwagę przy wyborze opcji raportu, ponieważ w rezultacie można wygenerować wiele komunikatów raportu.
- 2. Podczas umieszczania komunikatu w temacie może istnieć wiele kolejek docelowych, które mają zostać nadane kopii komunikatu. Jeśli niektóre z tych kolejek docelowych mają problem, na przykład zapełnianie kolejki, pomyślne zakończenie operacji MQPUT jest uzależnione od ustawienia wartości NPMGDLV lub PMGDLV (w zależności od trwałości komunikatu). Jeśli to ustawienie jest takie, że dostarczenie komunikatu do kolejki docelowej musi być pomyślne (na przykład jest to komunikat trwały dla trwałego subskrybenta, a parametr PMGDLV jest ustawiony na ALL lub ALLDUR), powodzenie jest definiowane jako jedno z następujących kryteriów:
 - Pomyślnie umieszczono w kolejce subskrybenta
 - Użycie obiektu RODLQ i pomyślne umieszczenie w kolejce niedostarczanych komunikatów, jeśli kolejka subskrybenta nie może odebrać komunikatu.
 - Użycie RODISC, jeśli kolejka subskrybenta nie może odebrać komunikatu.

Komunikaty raportów dla segmentów komunikatów:

- 1. Komunikaty raportów mogą być wymagane dla komunikatów, których segmentacja jest dozwolona (patrz opis flagi MFSEGA). Jeśli menedżer kolejek stwierdzi, że konieczne jest segmentowanie komunikatu, może zostać wygenerowany komunikat raportu dla każdego z segmentów, które następnie napotka odpowiedni warunek. Dlatego aplikacje powinny być przygotowane do odbierania wielu komunikatów raportów dla każdego typu żadanego komunikatu raportu. Pole *MDGID* w komunikacie raportu może być używane do korelowania wielu raportów z identyfikatorem

grupy oryginalnego komunikatu, a polem *MDFB* używanym do identyfikowania typu każdego komunikatu raportu.

2. Jeśli program GMLOGO jest używany do pobierania komunikatów raportu dla segmentów, należy pamiętać, że raporty o *różnych typach* mogą być zwracane przez kolejne wywołania MQGET. Na przykład, jeśli żądane są zarówno raporty COA, jak i COD dla komunikatu posegmentowanego przez menedżer kolejek, wywołania MQGET dla komunikatów raportu mogą zwrócić komunikaty raportu COA i COD interleaved w nieprzewidywalny sposób. Można tego uniknąć, korzystając z opcji GMCMPM (opcjonalnie z GMATM). Program GMCMPM powoduje, że menedżer kolejek składa ponownie komunikaty raportu, które mają ten sam typ raportu. Na przykład pierwsze wywołanie MQGET może ponownie złożyć wszystkie komunikaty COA odnoszące się do oryginalnego komunikatu, a drugie wywołanie MQGET może ponownie złożyć wszystkie komunikaty COD. To, które jest ponownie montowane, zależy od typu komunikatu, który ma być wyświetlany jako pierwszy w kolejce.
3. Aplikacje, które samodzielnie umieszczają segmenty, mogą określać różne opcje raportów dla każdego segmentu. Należy jednak zwrócić uwagę na następujące kwestie:
 - Jeśli segmenty są pobierane za pomocą opcji GMCMPM, tylko opcje raportu w segmencie *pierwszy* są honorowane przez menedżer kolejek.
 - Jeśli segmenty są pobierane jeden po jednym, a większość z nich ma jedną z opcji ROCOD*, ale co najmniej jeden segment nie, nie będzie możliwe użycie opcji GMCMPM w celu pobrania komunikatów raportu za pomocą pojedynczego wywołania MQGET lub użycie opcji GMASGA w celu wykrycia, kiedy wszystkie komunikaty raportu zostały wysłane.
4. W przypadku sieci MQ menedżery kolejek mogą mieć różne możliwości. Jeśli komunikat raportu dla segmentu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji, menedżer kolejek lub agent MCA nie będą domyślnie zawierały informacji o segmentach znajdujących się w komunikacie raportu, co może utrudnić zidentyfikowanie oryginalnego komunikatu, który spowodował wygenerowanie raportu. Tego typu trudności można uniknąć, żądając danych za pomocą komunikatu raportu, tj. poprzez określenie odpowiednich opcji RO* D lub RO* F. Należy jednak pamiętać, że jeśli określono RO* D, do aplikacji, która pobiera komunikat raportu, może zostać zwróconych *mniej niż* 100 bajtów danych komunikatu aplikacji, jeśli komunikat raportu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji.

Treść deskryptora komunikatu dla komunikatu raportu: Gdy menedżer kolejek lub agent kanału komunikatów (MCA) generuje komunikat raportu, ustawia pola w deskrytorze komunikatu na następujące wartości, a następnie umieszcza komunikat w normalny sposób.

Pole w strukturze MQMD	Użyta wartość
MDSID	MDSIDV
MDVER	MDVER2
MDREP	RONONE
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	W zależności od rodzaju raportu (FBCOA, FBCOD, FBEXP, lub wartość RC*)
MDENC	Skopiowano z oryginalnego deskryptora komunikatu
MDCSI	Skopiowano z oryginalnego deskryptora komunikatu
MDFMT	Skopiowano z oryginalnego deskryptora komunikatu
MDPRI	Skopiowano z oryginalnego deskryptora komunikatu
MDPER	Skopiowano z oryginalnego deskryptora komunikatu

Pole w strukturze MQMD	Użyta wartość
<i>MDMID</i>	Jak określono w opcjach raportu w oryginalnym deskrypcorze komunikatu
<i>MDCID</i>	Jak określono w opcjach raportu w oryginalnym deskrypcorze komunikatu
<i>MDBOC</i>	0
<i>MDRQ</i>	Puste
<i>MDRM</i>	Nazwa menedżera kolejek.
<i>MDUID</i>	Zgodnie z opcją PMPASI
<i>MDACC</i>	Zgodnie z opcją PMPASI
<i>MDAID</i>	Zgodnie z opcją PMPASI
<i>MDPAT</i>	ATQM lub jako odpowiedni dla agenta kanału komunikatów
<i>MDPAN</i>	Pierwsze 28 bajtów nazwy menedżera kolejek lub nazwy agenta kanału komunikatów. W przypadku komunikatów raportu wygenerowanych przez most IMS pole to zawiera nazwę grupy XCF i nazwę elementu XCF systemu IMS , do którego odnosi się komunikat.
<i>MDPD</i>	Data wysłania komunikatu raportu
<i>MDPT</i>	Czas wysłania komunikatu raportu
<i>MDAOD</i>	Puste
<i>MDGID</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MDSEQ</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MDOFF</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MDMFL</i>	Skopiowano z oryginalnego deskryptora komunikatu
<i>MDOLN</i>	Skopiowano z oryginalnego deskryptora komunikatu, jeśli nie jest OLUNDF, i ustaw na długość pierwotnych danych komunikatu w inny sposób

Zaleca się, aby aplikacja generującą raport ustawiała podobne wartości, z wyjątkiem następujących:

- Pole *MDRM* można ustawić na puste (menedżer kolejek zmieni to nazwę na nazwę lokalnego menedżera kolejek po umieszczeniu w nim komunikacie).
- Pola kontekstu powinny być ustawione przy użyciu opcji, która zostałaby użyta w odpowiedzi, zwykle PMPASI.

Analizowanie pola raportu: pole *MDREP* zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu zażądał konkretnego raportu, powinny korzystać z jednej z technik opisanych w sekcji [“Analizowanie pola raportu”](#) na stronie 1463.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest RONONE.

MDRM (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek odpowiedzi.

Jest to nazwa menedżera kolejek, do którego ma zostać wysłany komunikat odpowiedzi lub komunikat raportu. *MDRQ* to nazwa lokalna kolejki, która jest zdefiniowana w tym menedżerze kolejek.

Jeśli pole *MDRM* jest puste, lokalny menedżer kolejek wyszuka nazwę *MDRQ* w definicjach kolejek.

Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *MDRM* w przesyłanej komunikacie

jest zastępowana wartością atrybutu *RemoteQMgrName* z definicji kolejki zdalnej, a wartość ta zostanie zwrócona w deskrytorze komunikatu, gdy aplikacja odbierający wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, *MDRM*, która jest przesyłana z komunikatem, jest nazwą lokalnego menedżera kolejek.

Jeśli nazwa jest określona, może ona zawierać odstępy końcowe; pierwszy znak o kodzie zero i znaki następujące po nim są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane żadne sprawdzenie, że nazwa spełnia reguły nazewnictwa dla menedżerów kolejek lub że ta nazwa jest znana wysyłającemu menedżerowi kolejek. Jest to również prawda dla podanej nazwy, jeśli *MDRM* jest zastępowana w przekazanej wiadomości.

Jeśli kolejka zwrotna nie jest wymagana, zaleca się (choć nie jest to zaznaczone) pole *MDRM* powinno być puste; pole nie powinno być pozostawione niezainicjowane.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnioną spacjami do długości pola.

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Długość tego pola jest podana przez LNQM. Początkowa wartość tego pola to 48 znaków odstępu.

MDRQ (48-bajtowy łańcuch znaków)

Nazwa kolejki odpowiedzi.

Jest to nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania dla komunikatu, powinna wysłać komunikaty MTRPLY i MTRPRT. Nazwa to lokalna nazwa kolejki zdefiniowana w menedżerze kolejek identyfikowana przez produkt *MDRM*. Ta kolejka nie powinna być kolejką modelową, chociaż wysyłający menedżer kolejek nie weryfikuje tej kolejki po umieszczeniu w niej komunikacie.

W przypadku wywołań MQPUT i MQPUT1 pole to nie może być puste, jeśli pole *MDMT* ma wartość MTRQST lub jeśli żądane są komunikaty raportu w polu *MDREP* (Typ obiektu). Jednak podana wartość (lub podstawiona) jest przekazywana do aplikacji, która wydaje żądanie pobrania dla komunikatu, niezależnie od typu komunikatu.

Jeśli pole *MDRM* jest puste, lokalny menedżer kolejek będzie wyszukiwać nazwę *MDRQ* we własnych definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *MDRQ* w przesyłanej komunikacie jest zastępowana wartością atrybutu *RemoteQName* z definicji kolejki zdalnej, a wartość ta zostanie zwrócona w deskrytorze komunikatu, gdy aplikacja odbierający wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, *MDRQ* pozostaje niezmieniona.

Jeśli nazwa jest określona, może ona zawierać odstępy końcowe; pierwszy znak o kodzie zero i znaki następujące po nim są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane żadne sprawdzenie, że nazwa spełnia reguły nazewnictwa dla kolejek. Jest to również prawdziwe w przypadku nadawanej nazwy, jeśli *MDRQ* jest zastępowana w przesyłanej wiadomości. Jedynym sprawdzany jest fakt, że podano nazwę, jeśli wymagają tego okoliczności.

Jeśli kolejka zwrotna nie jest wymagana, zaleca się (choć nie jest to zaznaczone) pole *MDRQ* powinno być puste; pole nie powinno być pozostawione niezainicjowane.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnioną spacjami do długości pola.

Jeśli komunikat, który wymaga komunikatu raportu, nie może zostać dostarczony, a komunikat raportu również nie może zostać dostarczony do określonej kolejki, to zarówno oryginalny komunikat, jak i komunikat raportu są wyświetlane w kolejce niedostarczonych komunikatów (patrz atrybut *DeadLetterQName* opisany w sekcji "Atrybuty dla menedżera kolejek" na stronie 1427).

Jest to pole wyjściowe dla wywołania MQGET oraz pole wejściowe dla wywołań MQPUT i MQPUT1. Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

MDSEQ (10-cyfrowa liczba całkowita ze znakiem)

Numer kolejny komunikatu logicznego w grupie.

Numery kolejne rozpoczynają się od 1, a następnie zwiększają się o 1 dla każdego nowego komunikatu logicznego w grupie, do 999 999 999. Komunikat fizyczny, który nie znajduje się w grupie, ma numer kolejny 1.

To pole nie musi być ustawione przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET wartość MOSEQN *nie* jest określona.

Poniżej przedstawiono zalecane sposoby korzystania z tych wywołań w przypadku komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołania to MQPUT1, aplikacja musi upewnić się, że parametr MDSEQ jest ustawiony na odpowiednią wartość.

W przypadku danych wejściowych wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w Tabeli 1. W przypadku danych wyjściowych z wywołań MQPUT i MQPUT1 menedżer kolejek ustawia to pole na wartość, która została wysłana z komunikatem.

W przypadku danych wejściowych wywołania MQGET menedżer kolejek używa wartości określonej szczegółowo w Tabeli 1. W przypadku danych wyjściowych wywołania MQGET menedżer kolejek ustawia to pole na wartość dla pobranego komunikatu.

Wartość początkowa tego pola jest jedną z wartości. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

MDSIDV

Identyfikator struktury deskryptora komunikatu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MDSIDV.

MDUID (12-bajtowy łańcuch znaków)

Identyfikator użytkownika.

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

MDUID określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku.

Po odebraniu komunikatu MDUID można użyć w polu ODAU parametru OBJDSC kolejnego wywołania MQOPEN lub MQPUT1, tak aby sprawdzenie autoryzacji było wykonywane dla użytkownika produktu MDUID zamiast aplikacji wykonujących otwarcie.

Gdy menedżer kolejek generuje te informacje dla wywołania MQPUT lub MQPUT1, menedżer kolejek używa identyfikatora użytkownika określonego na podstawie środowiska.

Gdy identyfikator użytkownika jest określany na podstawie środowiska:

- W systemie z/OS menedżer kolejek używa:
 - W przypadku zadania wsadowego identyfikator użytkownika z karty zadania JES lub uruchomionego zadania
 - Dla TSO, identyfikator logowania użytkownika
 - W przypadku systemu CICS identyfikator użytkownika powiązany z zadaniem
 - W przypadku systemu IMS identyfikator użytkownika zależy od typu aplikacji:
 - Przez:
 - Regiony BMP niezwiązane z komunikatem
 - Niekomunikat regionów IFP
 - Komunikat BMP i regiony IFP komunikatu, które *nie* wydały pomyślnego wywołania GU

Menedżer kolejek używa identyfikatora użytkownika z karty JES regionu JES lub identyfikatora użytkownika TSO. Jeśli są one puste lub mają wartość NULL, używa nazwy bloku specyfikacji programu (PSB).

- Przez:

- Komunikaty BMP i regiony IFP komunikatu, które *mają* , wydały pomyślne wywołanie GU
- Regiony MPP

menedżer kolejek używa jednego z następujących elementów:

- Identyfikator zalogowanego użytkownika powiązany z komunikatem
 - Nazwa terminalu logicznego (LTERM)
 - Identyfikator użytkownika z karty pracy regionu JES
 - Identyfikator użytkownika TSO
 - Nazwa PSB
- W systemie IBM imenedżer kolejek używa nazwy profilu użytkownika powiązanego z zadaniem aplikacji.
 - W systemie HP Integrity NonStop Servermenedżer kolejek używa nazwy użytkownika MQSeries , która jest zdefiniowana dla identyfikatora użytkownika Tandem w głównej bazie danych produktu MQSeries .
 - W systemie Systemy UNIXmenedżer kolejek używa:
 - Nazwa logowania aplikacji
 - Efektywny identyfikator użytkownika procesu, jeśli logowanie nie jest dostępne
 - Identyfikator użytkownika powiązany z transakcją, jeśli aplikacja jest transakcją produktu CICS .
 - W systemie VSE/ESAjest to pole zastrzeżone.
 - W systemie Windowsmenedżer kolejek korzysta z pierwszych 12 znaków nazwy zalogowanego użytkownika.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze *PMO* określono PMSETI lub PMSETA. Wszystkie informacje znajdujące się po znaku NULL w tym polu są usuwane. Znak o kodzie zero i wszystkie następujące znaki są przekształcane w puste miejsca przez menedżer kolejek. Jeśli wartość PMSETI lub PMSETA nie została określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 w tym polu znajduje się *MDUID* , który został przesłany z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość *MDUID* przechowywana razem z komunikatem, jeśli zostanie zachowana (patrz opis PMRET, aby uzyskać więcej informacji na temat zachowanych publikacji), ale nie jest ona używana jako *MDUID* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępniają wartość do nadpisania *MDUID* we wszystkich publikacjach wysyłanych do nich. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

To jest pole wyjściowe dla wywołania MQGET. Długość tego pola jest podana przez LNUID. Początkowa wartość tego pola to 12 znaków odstępu.

MDVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

MDVER1

Struktura deskryptora komunikatu Version-1 .

MDVER2

Struktura deskryptora komunikatu Version-2 .

Uwaga: Gdy używany jest deskryptor MQMD w wersji version-2 , menedżer kolejek wykonuje dodatkowe sprawdzenia wszystkich struktur nagłówka produktu MQ , które mogą być obecne na początku danych komunikatu aplikacji. Szczegółowe informacje na ten temat zawiera uwagi dotyczące składni wywołania MQPUT.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MDVERC

Bieżąca wersja struktury deskryptora komunikatu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MDVER1.

Wartości początkowe

Tabela 148. Początkowe wartości pól w MQMD		
Nazwa pola	Nazwa stałej	Wartość stałej
MDSID	MDSIDV	'MD--'
MDVER	MDVER1	1
MDREP	RONONE	0
MDMT	MTDGRM	8
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT	Zależy od środowiska
MDCSI	CSQM	0
MDFMT	FMNONE	Puste
MDPRI	PRQDEF	-1
MDPER	PEQDEF	2
MDMID	MINONE	Wartości null
MDCID	CINONE	Wartości null
MDBOC	Brak	0
MDRQ	Brak	Puste
MDRM	Brak	Puste
MDUID	Brak	Puste
MDACC	ACNONE	Wartości null
MDAID	Brak	Puste
MDPAT	ATNCON	0
MDPAN	Brak	Puste
MDPD	Brak	Puste
MDPT	Brak	Puste
MDAOD	Brak	Puste
MDGID	GINONE	Wartości null
MDSEQ	Brak	1

Tabela 148. Początkowe wartości pól w MQMD (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
MDOFF	Brak	0
MDMFL	MFBRK	0
MDOLN	OLUNDF	-1
Uwagi: 1. Symbol – reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQMD Structure
D*
D* Structure identifier
D MDSID          1      4    INZ('MD ')
D* Structure version number
D MDVER          5      8I 0 INZ(1)
D* Options for report messages
D MDREP          9     12I 0 INZ(0)
D* Message type
D MDMT          13     16I 0 INZ(8)
D* Message lifetime
D MDEXP         17     20I 0 INZ(-1)
D* Feedback or reason code
D MDFB          21     24I 0 INZ(0)
D* Numeric encoding of message data
D MDENC         25     28I 0 INZ(273)
D* Character set identifier of messagedata
D MDCSI         29     32I 0 INZ(0)
D* Format name of message data
D MDFMT         33     40    INZ(' ')
D* Message priority
D MDPRI         41     44I 0 INZ(-1)
D* Message persistence
D MDPER         45     48I 0 INZ(2)
D* Message identifier
D MDMID         49     72    INZ(X'00000000000000-
D                      00000000000000000000-
D                      000000000000')
D* Correlation identifier
D MDCID         73     96    INZ(X'00000000000000-
D                      00000000000000000000-
D                      000000000000')
D* Backout counter
D MDBOC         97    100I 0 INZ(0)
D* Name of reply queue
D MDRQ         101    148    INZ
D* Name of reply queue manager
D MDRM         149    196    INZ
D* User identifier
D MDUID        197    208    INZ
D* Accounting token
D MDACC        209    240    INZ(X'00000000000000-
D                      00000000000000000000-
D                      00000000000000000000-
D                      000000')
D* Application data relating to identity
D MDAID        241    272    INZ
D* Type of application that put the message
D MDPAT        273    276I 0 INZ(0)
D* Name of application that put the message
D MDPAN        277    304    INZ
D* Date when message was put
D MDPD        305    312    INZ
D* Time when message was put
D MDPT        313    320    INZ
D* Application data relating to origin
D MDAOD        321    324    INZ
D* Group identifier

```

D	MDGID	325	348	INZ(X'0000000000000000-
D				00000000000000000000-
D				000000000000')
D*	Sequence number of logical message	within group		
D	MDSEQ	349	352I 0	INZ(1)
D*	Offset of data in physical message	from start of logical message		
D	MDOFF	353	356I 0	INZ(0)
D*	Message flags			
D	MDMFL	357	360I 0	INZ(0)
D*	Length of original message			
D	MDOLN	361	364I 0	INZ(-1)

MQMDE-Rozszerzenie deskryptora komunikatu

Przegląd

Cel: Struktura MQMDE opisuje dane, które czasami występują przed danymi komunikatu aplikacji. Struktura zawiera te pola MQMD, które istnieją w MQMD w wersji version-2, ale nie w przypadku deskryptora MQMD w wersji version-1.

Nazwa formatu: FMMDE.

Zestaw znaków i kodowanie: dane w produkcie MQMDE muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQMDE muszą być ustawione w polach *MDCSI* i *MDENC* w:

- MQMD (jeśli struktura MQMDE znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQMDE (wszystkie inne obserwacje).

Jeśli tabela MQMDE nie znajduje się w zestawie znaków i kodowaniu menedżera kolejek, wartość MQMDE jest akceptowana, ale nie została uhonorowana, oznacza to, że MQMDE jest traktowane jako dane komunikatu.

Użycie: zwykłe aplikacje powinny używać deskryptora MQMD z wersji version-2, w którym to przypadku nie będą one napotykały struktury MQMDE. Jednak wyspecjalizowane aplikacje i aplikacje, które w dalszym ciągu używają deskryptora MQMD z wersji version-1, mogą w niektórych sytuacjach napotkać MQMDE. Struktura MQMDE może wystąpić w następujących okolicznościach:

- Określone w wywołaniach MQPUT i MQPUT1
- Zwrócone przez wywołanie MQGET
- W komunikatach w kolejkach transmisji
- [“Określono MQMDE w wywołaniach MQPUT i MQPUT1” na stronie 1170](#)
- [“MQMDE zwrócone przez wywołanie MQGET” na stronie 1171](#)
- [“MQMDE w komunikatach w kolejkach transmisji” na stronie 1171](#)
- [“Pola” na stronie 1172](#)
- [“Wartości początkowe” na stronie 1174](#)
- [“Deklaracja RPG” na stronie 1174](#)

Określono MQMDE w wywołaniach MQPUT i MQPUT1

W wywołaniach MQPUT i MQPUT1, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, aplikacja może opcjonalnie prefikować dane komunikatu za pomocą MQMDE, ustawiając pole *MDFMT* w MQMD na FMMDE, aby wskazać, że jest obecna MQMDE. Jeśli aplikacja nie udostępnia wywołania MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w MQMDE. Wartości domyślne używane przez menedżera kolejek są takie same, jak początkowe wartości struktury-patrz [Tabela 150 na stronie 1174](#).

If the application provides a version-2 MQMD oraz prefixes the application message data with an MQMDE, the structures are processed as shown in [Tabela 149 na stronie 1171](#).

Tabela 149. Działanie menedżera kolejek po określeniu MQMDE w MQPUT lub MQPUT1			
Wersja MQMD	Wartości w polach version-2	Wartości odpowiednich pól w MQMDE	Działanie podjęte przez menedżera kolejek
1	-	Ważne	MQMDE jest honorowany
2	Domyślny	Ważne	MQMDE jest honorowany
2	Niedomyślna	Ważne	MQMDE jest traktowane jako dane komunikatu
1 lub 2	Dowolna	Niepoprawne	Wywołanie nie powiodło się z odpowiednim kodem przyczyny
1 lub 2	Dowolna	MQMDE znajduje się w niewłaściwym zestawie znaków lub kodowaniu, albo jest nieobsługiwaną wersją	MQMDE jest traktowane jako dane komunikatu

Jest jedna szczególna sprawa. Jeśli aplikacja używa deskryptora MQMD z wersji version-2 w celu umieszczenia komunikatu, który jest segmentem (czyli ustawiono flagę MFSEG lub MFLSEG), a nazwa formatu w strukturze MQMD to FMDLH, menedżer kolejek generuje strukturę MQMDE i wstawia ją *między* strukturą MQDLH a danymi, które są zgodne z tą strukturą. W deskryptywie MQMD, który menedżer kolejek zachowuje z komunikatem, pola version-2 są ustawiane na wartości domyślne.

Kilka pól istniejących w strukturze MQMD version-2, ale nie version-1 MQMD, są polami wejściowymi/ wyjściowymi w tabelach MQPUT i MQPUT1. Jednak menedżer kolejek *nie* zwraca żadnych wartości w równoważnych polach w MQMDE na danych wyjściowych z wywołań MQPUT i MQPUT1. Jeśli aplikacja wymaga tych wartości wyjściowych, musi ona używać deskryptora MQMD z wersji version-2.

MQMDE zwrócone przez wywołanie MQGET

W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, menedżer kolejek prefiksuje komunikat zwrócony przez produkt MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość niedomyślną. Menedżer kolejek ustawia pole *MDFMT* w deskryptywie MQMD na wartość FMMDE, aby wskazać, że jest obecna MQMDE.

Jeśli aplikacja udostępnia parametr MQMDE na początku parametru *BUFFER*, MQMDE jest ignorowane. W przypadku powrotu z wywołania MQGET jest on zastępowany przez MQMDE dla komunikatu (jeśli jest wymagany) lub nadpisywany przez dane komunikatu aplikacji (jeśli MQMDE nie jest potrzebne).

Jeśli wywołanie MQMDE jest zwracane przez wywołanie MQMDE, dane w MQMDE są zwykle w zestawie znaków i kodowaniu menedżera kolejek. Jednak MQMDE może znajdować się w innym zestawie znaków i kodowaniu, jeśli:

- Produkt MQMDE został potraktowany jako dane w wywołaniu MQPUT lub MQPUT1 (patrz [Tabela 149 na stronie 1171](#) w celu uzyskania informacji o okolicznościach, które mogą być przyczyną tego wywołania).
- Komunikat został odebrany ze zdalnego menedżera kolejek połączonego przez połączenie TCP, a odbierający agent kanału komunikatów (MCA) nie został poprawnie skonfigurowany (więcej informacji na ten temat zawiera sekcja [Zabezpieczenia obiektów produktu IBM MQ for IBM i](#)).

MQMDE w komunikatach w kolejkach transmisji

Komunikaty w kolejkach transmisji są poprzedzane strukturą MQXQH, która zawiera w sobie deskryptor MQMD version-1. Może być również obecny produkt MQMDE, który znajduje się między strukturą MQXQH a danymi komunikatu aplikacji, ale będzie on zazwyczaj obecny tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość niedomyślną.

Między strukturą MQXQH a danymi komunikatu aplikacji mogą występować również inne struktury nagłówka produktu IBM MQ. Na przykład, gdy występuje nagłówek niedostarczonych komunikatów MQDLH, a komunikat nie jest segmentem, to kolejność jest następująca:

- MQXQH (zawiera element MQMD w wersji version-1)
- MQMDE
- MQDLH
- Dane komunikatu aplikacji

Pola

Struktura MQMDE zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

MECSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych, który jest zgodny z MQMDE.

Określa identyfikator zestawu znaków dla danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych znakowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Można użyć następującej wartości specjalnej:

CINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest *ATBRKR*.

Wartością początkową tego pola jest CSUNDF.

MEENC (10-cyfrowa liczba całkowita ze znakiem)

MEENC (10-cyfrowa liczba całkowita ze znakiem)

Określa kodowanie liczbowe dla danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych liczbowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Więcej informacji na temat kodowania danych można znaleźć w sekcji *MDENC* opisanej w sekcji [“MQMD-deskryptor komunikatu” na stronie 1124](#).

Wartością początkową tego pola jest ENNAT.

MEFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi ogólne.

Można określić następującą opcję:

MEFNON

Brak flag.

Wartością początkową tego pola jest MEFNON.

MEFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych, które są następujące: MQMDE.

Określa nazwę formatu danych, które są zgodne ze strukturą MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Więcej informacji na temat nazw formatów znajduje się w sekcji *MDFMT* opisanej w sekcji “MQMD-deskryptor komunikatu” na stronie 1124 .

Wartością początkową tego pola jest FMNONE.

MEGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Zapoznaj się z polem *MDGID* , które opisano w sekcji “MQMD-deskryptor komunikatu” na stronie 1124. Wartością początkową tego pola jest GINONE.

MELEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQMDE.

Zdefiniowana jest następująca wartość:

MELEN2

Długość struktury rozszerzenia deskryptora komunikatu version-2 .

Początkowa wartość tego pola to MELEN2.

MEMFL (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatu.

Zapoznaj się z polem *MDMFL* , które opisano w sekcji “MQMD-deskryptor komunikatu” na stronie 1124. Wartością początkową tego pola jest MFNONE.

MEOFF (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

Zapoznaj się z polem *MDOFF* , które opisano w sekcji “MQMD-deskryptor komunikatu” na stronie 1124. Wartością początkową tego pola jest 0.

MEOLN (10-cyfrowa liczba całkowita ze znakiem)

Długość oryginalnego komunikatu.

Zapoznaj się z polem *MDOLN* , które opisano w sekcji “MQMD-deskryptor komunikatu” na stronie 1124. Wartością początkową tego pola jest OLUNDF.

MESEQ (10-cyfrowa liczba całkowita ze znakiem)

Numer kolejny komunikatu logicznego w grupie.

Zapoznaj się z polem *MDSEQ* , które opisano w sekcji “MQMD-deskryptor komunikatu” na stronie 1124. Wartością początkową tego pola jest 1.

MESID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

MESIDV

Identyfikator struktury rozszerzenia deskryptora komunikatu.

Wartością początkową tego pola jest MESIDV.

MEVER (dziesięciocyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

MEVER2

Struktura rozszerzenia deskryptora komunikatu Version-2 .

Następująca stała określa numer wersji bieżącej wersji:

MEVERC

Bieżąca wersja struktury rozszerzenia deskryptora komunikatu.

Początkowa wartość tego pola to MEVER2.

Wartości początkowe

Tabela 150. Początkowe wartości pól w MQMDE		
Nazwa pola	Nazwa stałej	Wartość stałej
MESID	MESIDV	'MDE¬'
MEVER	MEVER2	2
MELEN	MELEN2	72
MEENC	ENNAT	Zależy od środowiska
MECSI	CSUNDF	0
MEFMT	FMNONE	Puste
MEFLG	MEFNON	0
MEGID	GINONE	Wartości null
MESEQ	Brak	1
MEOFF	Brak	0
MEMFL	MFBRK	0
MEOLN	OLUNDF	-1
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQMDE Structure
D*
D* Structure identifier
D MESID          1      4      INZ('MDE ')
D* Structure version number
D MEVER          5      8I 0 INZ(2)
D* Length of MQMDE structure
D MELEN          9     12I 0 INZ(72)
D* Numeric encoding of data that followsMQMDE
D MEENC         13     16I 0 INZ(273)
D* Character-set identifier of data thatfollows MQMDE
D MECSI         17     20I 0 INZ(0)
D* Format name of data that followsMQMDE
D MEFMT         21     28      INZ('      ')
D* General flags
D MEFLG         29     32I 0 INZ(0)
D* Group identifier
D MEGID         33     56      INZ(X'00000000000000-
D                      00000000000000000000-
D                      000000000000')
D* Sequence number of logical messagewithin group
D MESEQ         57     60I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
D MEOFF         61     64I 0 INZ(0)
D* Message flags
D MEMFL         65     68I 0 INZ(0)
D* Length of original message
D MEOLN         69     72I 0 INZ(-1)
```

MQMHBO-uchwyt komunikatu do opcji buforu

Struktura definiująca uchwyt komunikatu do opcji buforu

Przegląd

Przeznaczenie: Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki bufory są generowane z uchwytów komunikatów. Struktura jest parametrem wejściowym w wywołaniu MQMHBUF.

Zestaw znaków i kodowanie: Dane w MQMHBO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1175](#)
- [“Wartości początkowe” na stronie 1176](#)
- [“Deklaracja RPG” na stronie 1176](#)

Pola

Struktura MQMHBO zawiera następujące pola: pola są opisane w **kolejności alfabetycznej**:

MBOPT (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt komunikatu do struktury opcji buforu-pole MBOPT.

Te opcje sterują działaniem MQMHBUF.

Należy podać następującą opcję:

MBPRRF

Przekształcając właściwości z uchwytu komunikatu w bufor, przekształć je w format MQRFH2 .

Opcjonalnie można również określić następującą opcję. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

MBDLPR

Właściwości, które są dodawane do buforu, są usuwane z uchwytu komunikatu. Jeśli wywołanie nie powiedzie się, żadne właściwości nie zostaną usunięte.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest MBPRRF.

MBSID (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt komunikatu do struktury opcji buforu-pole MBSID.

Jest to identyfikator struktury. Wartość musi być następująca:

MBSIDV

Identyfikator uchwytu komunikatu do struktury opcji buforu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola isMBSIDV.

MBVER (dziesięciocyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

MBVER1

Numer wersji dla uchwytu komunikatu do struktury opcji buforu.

Następująca stała określa numer wersji bieżącej wersji:

MBVERC

Bieżąca wersja uchwytu komunikatu do struktury opcji buforu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to MBVER1.

Wartości początkowe

Tabela 151. Początkowe wartości pól w MQMHBO		
Nazwa pola	Nazwa stałej	Wartość stałej
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1
MBOPT	MBPRRF	

Uwagi:

1. Łańcuch wartości NULL lub puste znaki oznaczają pusty znak.

Deklaracja RPG

```
D* MQMHBO Structure
D*
D*
D* Structure identifier
D MBSID 1 4 INZ('MHBO')
D*
D* Structure version number
D MBVER 5 8I 0 INZ(1)
D*
D* Options that control the action of MQMHBUF
D MBOPT 9 12I 0 INZ(1)
```

MQOD-deskryptor obiektu

Struktura MQOD służy do określania obiektu według nazwy.

Przegląd

Cel: Następujące typy obiektów są poprawne:

- Kolejka lub lista dystrybucyjna
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat

Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQOPEN i MQPUT1.

Wersja: Bieżąca wersja programu MQOD to ODVER4. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję programu MQOD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *ODVER* ustawioną na ODVER1. Aby użyć pól, które nie są obecne w strukturze version-1, aplikacja musi ustawić pole *ODVER* na numer wersji wymaganej wersji.

Aby otworzyć listę dystrybucyjną, *ODVER* musi mieć wartość ODVER2 lub większą.

Zestaw znaków i kodowanie: Dane w tabeli MQOD muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1177](#)
- [“Wartości początkowe” na stronie 1184](#)
- [“Deklaracja RPG” na stronie 1185](#)

Pola

Struktura MQOD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ODASI (40-bajtowy łańcuch bitowy)

Alternatywny identyfikator zabezpieczeń.

Jest to identyfikator zabezpieczeń przekazywany razem z produktem *ODAU* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji. Produkt *ODASI* jest używany tylko wtedy, gdy:

- Wartość *OOALTU* jest określona w wywołaniu *MQOPEN*, lub
- Parametr *PMALTU* jest określony w wywołaniu *MQPUT1*,

Pole *i* pole *ODAU* nie jest całkowicie puste w stosunku do pierwszego znaku o kodzie zero lub do końca pola.

Pole *ODASI* ma następującą strukturę:

- Pierwszy bajt jest binarną liczbą całkowitą zawierającą długość znaczących danych, które są następujące; wartość nie obejmuje samego bajtu o długości. Jeśli identyfikator zabezpieczeń nie jest obecny, długość wynosi zero.
- Drugi bajt wskazuje typ identyfikatora zabezpieczeń, który jest obecny. Możliwe są następujące wartości:

SITWNT

Identyfikator zabezpieczeń produktu Windows .

SITNON

Brak identyfikatora zabezpieczeń.

- Trzeci i kolejne bajty aż do długości zdefiniowanej przez pierwszy bajt zawierają sam identyfikator zabezpieczeń.
- Pozostałe bajty w polu są ustawione na zero binarne.

Można użyć następującej wartości specjalnej:

SINONE

Nie określono identyfikatora zabezpieczeń.

Wartość jest binarna zero dla długości pola.

To jest pole wejściowe. Długość tego pola jest podana przez *LNSCID*. Wartością początkową tego pola jest *SINONE*. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER3*.

ODAU (12-bajtowy łańcuch znaków)

Alternatywny identyfikator użytkownika.

Jeśli wartość *OOALTU* została określona dla wywołania *MQOPEN* lub *PMALTU* dla wywołania *MQPUT1*, to pole zawiera alternatywny identyfikator użytkownika, który ma być używany do sprawdzania autoryzacji dla otwartego, w miejsce identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona. Niektóre operacje sprawdzania są jednak nadal wykonywane z bieżącym identyfikatorem użytkownika (na przykład sprawdzeniami kontekstowymi).

Jeśli nie określono wartości *OOALTU* i *PMALTU*, a pole to jest całkowicie puste, do pierwszego znaku o wartości *NULL* lub do końca pola, to otwarcie może zakończyć się powodzeniem tylko wtedy, gdy nie jest wymagane uprawnienie użytkownika do otwarcia tego obiektu przy użyciu podanych opcji.

Jeśli nie określono ani *OOALTU*, ani *PMALTU*, to pole jest ignorowane.

To jest pole wejściowe. Długość tego pola jest podana przez *LNUID*. Początkowa wartość tego pola to 12 znaków odstępu.

ODDN (48-bajtowy łańcuch znaków)

Nazwa kolejki dynamicznej.

Jest to nazwa kolejki dynamicznej, która ma zostać utworzona przy użyciu wywołania MQOPEN. Ma to znaczenie tylko wtedy, gdy parametr *ODON* określa nazwę kolejki modelowej; we wszystkich pozostałych przypadkach *ODDN* jest ignorowany.

Znaki, które są poprawne w nazwie, są takie same jak w przypadku produktu *ODON*, z tą różnicą, że znak gwiazdki jest również poprawny. Nazwa, która jest pusta (lub jedna, w której znaki puste są wyświetlane przed pierwszym znakiem o kodzie zero), nie jest poprawna, jeśli *ODON* jest nazwą kolejki modelowej.

Jeśli ostatni niepusty znak w nazwie to gwiazdka (*), to menedżer kolejek zastępuje gwiazdkę łańcuchem znaków, który gwarantuje, że nazwa wygenerowana dla kolejki jest unikalna w lokalnym menedżerze kolejek. Aby możliwe było użycie wystarczającej liczby znaków, gwiazdka jest poprawna tylko na pozycjach od 1 do 33. Po gwiazdce nie mogą występować znaki inne niż spacje lub znak o kodzie zero.

Jest ona poprawna, aby gwiazdka wystąpiła na pierwszej pozycji znaku, w którym to przypadku nazwa składa się wyłącznie z znaków wygenerowanych przez menedżer kolejek.

To jest pole wejściowe. Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 'AMQ.*', dopełniona odstępami.

ODIDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba kolejek, których otwarcie nie powiodło się.

Jest to liczba kolejek z listy dystrybucyjnej, które nie zostały pomyślnie otwarte. Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Uwaga: Jeśli jest obecny, to pole jest ustawione *tylko*, jeśli parametr *CMPCOD* w wywołaniu MQOPEN lub MQPUT1 ma wartość CCOK lub CCWARN; *nie* jest ustawiony, jeśli parametr *CMPCOD* ma wartość CCFAIL.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODKDC (10-cyfrowa liczba całkowita ze znakiem)

Pomyślnie otwarto liczbę kolejek lokalnych.

Jest to liczba kolejek znajdujących się na liście dystrybucyjnej, które są rozstrzygane do kolejek lokalnych i które zostały pomyślnie otwarte. Liczba ta nie obejmuje kolejek rozstrzyganych do kolejek zdalnych (nawet jeśli początkowo używana jest lokalna kolejka transmisji do przechowywania komunikatu). Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek obiektów.

Jest to nazwa menedżera kolejek, w którym zdefiniowany jest obiekt *ODON*. Znaki, które są poprawne w nazwie, są takie same jak w przypadku produktu *ODON* (patrz wcześniej). Nazwa, która jest całkowicie pusta, do pierwszego znaku o wartości NULL lub do końca pola oznacza menedżer kolejek, z którym połączona jest aplikacja (lokalny menedżer kolejek).

Następujące punkty mają zastosowanie do typów wskazanych obiektów:

- Jeśli *ODOT* to OTTOP, OTNLST, OTPRO lub OTQM, wartość *ODMN* musi być pusta lub musi być nazwą lokalnego menedżera kolejek.
- Jeśli *ODON* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej, a zwraca w polu *ODMN* nazwę menedżera kolejek, w którym tworzona jest kolejka. Jest to nazwa lokalnego menedżera kolejek. Kolejka modelowa może być określona tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1.

- Jeśli *ODON* jest nazwą kolejki klastra, a *ODMN* jest pusta, to rzeczywiste miejsce docelowe komunikatów wysyłanych za pomocą uchwytu kolejki zwracanego przez wywołanie MQOPEN jest wybierane przez menedżer kolejek (lub wyjście obciążenia klastra, jeśli jest zainstalowane) w następujący sposób:
 - Jeśli określono parametr OOBND0, menedżer kolejek wybiera instancję kolejki klastra podczas przetwarzania wywołania MQOPEN, a wszystkie komunikaty umieszczone przy użyciu tego uchwytu kolejki są wysyłane do tej instancji.
 - Jeśli określono OOBNDN, menedżer kolejek może wybrać inną instancję kolejki docelowej (rezydując w innym menedżerze kolejek w klastrze) dla każdej kolejnej wywołania MQPUT, które używa tego uchwytu kolejki.

Jeśli aplikacja musi wysłać komunikat do *konkretnej* instancji kolejki klastra (czyli instancji kolejki znajdującej się w określonym menedżerze kolejek w klastrze), aplikacja powinna określić nazwę tego menedżera kolejek w polu *ODMN*. Wymusza wysłanie komunikatu przez lokalny menedżer kolejek do określonego docelowego menedżera kolejek.

- Jeśli otwierany obiekt jest listą dystrybucyjną (to znaczy *ODREC* jest większa od zera), wartość *ODMN* musi być pusta lub zawierać łańcuch pusty. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2153.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ODON* jest nazwą kolejki modelowej, a pole tylko wejściowe we wszystkich innych przypadkach. Długość tego pola jest podana przez LNQM. Początkowa wartość tego pola to 48 znaków odstępu.

ODON (48-bajtowy łańcuch znaków)

Nazwa obiektu.

Jest to nazwa lokalna obiektu zdefiniowana w menedżerze kolejek identyfikowanego przez produkt *ODMN*. Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (A-Z)
- Małe litery alfabetu (a-z)
- Cyfry cyfry (0-9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Znak o kodzie zero może być używany do wskazywania końca istotnych danych w nazwie; wartości null i dowolnych po nim znaków są traktowane jako znaki puste. W środowiskach wskazanych poniżej obowiązują następujące ograniczenia:

- W systemach, w których używane jest kodowanie EBCDIC Katakana, nie można używać małych liter.
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w znaki cudzysłowu, gdy są określone w komendach. Tych znaków cudzysłowu nie należy określać dla nazw, które występują jako pola w strukturach lub jako parametry wywołań.

Następujące punkty mają zastosowanie do typów wskazanych obiektów:

- Jeśli *ODON* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej, a zwraca w polu *ODON* nazwę utworzonej kolejki. Kolejka modelowa może być określona tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1.
- Jeśli otwierany obiekt jest listą dystrybucyjną (to znaczy *ODREC* jest obecna i większa od zera), *ODON* musi być pusty lub łańcuch pusty. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2152.
- Jeśli *ODOT* to OTQM, mają zastosowanie reguły specjalne; w tym przypadku nazwa musi być całkowicie pusta, aż do pierwszego znaku o wartości NULL lub do końca pola.
- Jeśli wartością *ODON* jest nazwa kolejki aliasowej z typem TARGTYPE (TOPIC), najpierw w nazwanej kolejce aliasowej zostanie sprawdzona kontrola bezpieczeństwa, co jest normalne w przypadku użycia kolejek aliasowych. Jeśli to sprawdzenie zabezpieczeń zakończy się pomyślnie, to wywołanie

MQOPEN będzie kontynuowane i zachowuje się jak komenda MQOPEN dla obiektu OTTOP, w tym sprawdzenie zabezpieczeń względem obiektu tematu administracyjnego.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ODON* jest nazwą kolejki modelowej, a pole tylko wejściowe we wszystkich innych przypadkach. Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ODON* i *ODOS*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

ODORO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu obiektu od początku MQOD.

Jest to przesunięcie w bajtach pierwszego rekordu obiektu MQOR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. *ODORO* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *ODREC* wynosi zero.

Gdy lista dystrybucyjna jest otwierana, w celu określenia nazw kolejek docelowych na liście dystrybucyjnej należy podać tablicę jednej lub większej liczby rekordów obiektu MQOR. Można to zrobić na jeden z dwóch sposobów:

- Za pomocą pola przesunięcia *ODORO*

W takim przypadku aplikacja powinna zadeklarować własną strukturę zawierającą tabelę MQOD, po której następuje tablica rekordów MQOR (wraz z wymaganą wieloma elementami tablicy), a następnie ustawić *ODORO* na przesunięcie pierwszego elementu w tablicy od początku tabeli MQOD. Należy zwrócić uwagę, aby przesunięcie to było poprawne.

- Za pomocą pola wskaźnika *ODORP*

W takim przypadku aplikacja może zadeklarować tablicę struktur MQOR niezależnie od struktury MQOD, a następnie ustawić wartość *ODORP* na adres tablicy.

Niezależnie od wybranej techniki, należy użyć jednego z produktów *ODORO* i *ODORP*; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2155, jeśli oba są zerowe, albo oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODORP (wskaźnik)

Adres pierwszego rekordu obiektu.

Jest to adres pierwszego rekordu obiektu MQOR. *ODORP* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *ODREC* wynosi zero.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. *ODORP* lub *ODORO* mogą być używane do określania rekordów obiektów, ale nie obu; więcej informacji na ten temat zawiera opis pola *ODORO*. Jeśli produkt *ODORP* nie jest używany, należy go ustawić na pusty wskaźnik lub zerową liczbę bajtów. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODOS (MQCHARV)

ODOS określa długą nazwę obiektu, który ma być używany.

To pole jest przywoływane tylko dla określonych wartości *ODOT*. Patrz opis *ODOT*, aby uzyskać szczegółowe informacje o tym, które wartości wskazują, że to pole jest używane.

Jeśli wartość *ODOS* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2441.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze *MQCHARV*.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ODON* i *ODOS*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER4*.

ODOT (10-cyfrowa liczba całkowita ze znakiem)

Typ obiektu.

Typ obiektu nazwanego w *ODON*. Dozwolone są następujące wartości:

OTQ

do kolejki błędów. Nazwa obiektu znajduje się w *ODON*.

OTNLST

Lista nazw. Nazwa obiektu znajduje się w *ODON*.

OTPRO

Definicja procesu. Nazwa obiektu znajduje się w *ODON*.

OTQM

menedżerze kolejek. Nazwa obiektu znajduje się w *ODON*.

OTTOP

. Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *ODON* i *ODOS*.

Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

If the object identified by the *ODON* field cannot be found, the call will fail with reason code RC2425 even if there is a string specified in *ODOS*.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest OTQ.

ODREC (10-cyfrowa liczba całkowita ze znakiem)

Liczba obecnych rekordów obiektów.

Jest to liczba rekordów obiektów MQOR, które zostały udostępnione przez aplikację. Jeśli ta liczba jest większa od zera, oznacza to, że lista dystrybucyjna jest otwierana, a *ODREC* jest liczbą kolejek docelowych na liście. Jest ona poprawna dla listy dystrybucyjnej, która może zawierać tylko jedno miejsce docelowe.

Wartość parametru *ODREC* nie może być mniejsza niż zero, a jeśli jest większa niż zero *ODOT* musi być wartością OTQ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2154, jeśli warunki te nie są spełnione.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODRMN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa menedżera kolejek.

Jest to nazwa docelowego menedżera kolejek po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *ODRQN*. *ODRMN* może być nazwą lokalnego menedżera kolejek.

Jeśli *ODRQN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, *ODRMN* jest nazwą grupy współużytkowania kolejki. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, *ODRQN* może być nazwą grupy współużytkowania kolejki lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejki (rodzaj zwróconej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *ODRMN* jest pusty:

- To nie jest kolejka

- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.
- Kolejka klastra o podanej nazwie OOBNDN (lub z wartością OOBNDQ w przypadku, gdy atrybut kolejki *DefBind* ma wartość BNDNOT)
- Lista dystrybucyjna

To jest pole wyjściowe. Długość tego pola jest podana przez LNQN. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER3.

ODRO (MQCHARV)

ODRO to długa nazwa obiektu po przetłumaczonej nazwie menedżera kolejek w programie *ODON*.

To pole jest zwracane tylko w przypadku niektórych typów obiektów, tematów i aliasów kolejek, które odwołują się do obiektu tematu.

Jeśli długa nazwa obiektu jest dostępna w produkcie *ODOS*, a w produkcie *ODON* nie jest udostępniana żadna wartość, wartość zwrócona w tym polu jest taka sama, jak wartość podana w *ODOS*.

Jeśli to pole zostanie pominięte (to jest ODRO.VSBufSize ma wartość zero), wartość *ODRO* nie jest zwracana, ale długość jest zwracana w ODRO.VSLength. Jeśli długość jest krótsza niż pełny *ODRO*, to jest ona obcinana i zwraca tyle znaków z prawej strony, co może zmieścić się w podanej długości.

Jeśli wartość *ODRO* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2520. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER4.

ODRQN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa kolejki.

Jest to nazwa kolejki docelowej po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa kolejki, która istnieje w menedżerze kolejek identyfikowanego przez produkt *ODRMN*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *ODRQN* jest pusty:

- To nie jest kolejka
- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.
- Lista dystrybucyjna
- Kolejka aliasowa, która odwołuje się do obiektu tematu (zamiast tego należy zapoznać się z sekcją [“ODRO \(MQCHARV\)” na stronie 1182](#)).

To jest pole wyjściowe. Długość tego pola jest podana przez LNQN. Wartość początkowa tego pola jest łańcuchem pustym w języku C, a 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER3.

ODRRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu odpowiedzi od początku MQOD.

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. *ODRRO* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *ODREC* wynosi zero.

Po otwarciu listy dystrybucyjnej można podać tablicę jednego lub większej liczby rekordów odpowiedzi MQRR, aby zidentyfikować kolejki, których otwarcie nie powiodło się (pole *RRCC* w MQRR), oraz przyczynę każdego niepowodzenia (pole *RRREA* w tabeli MQRR). Dane są zwracane w tablicy rekordów odpowiedzi w tej samej kolejności, w jakiej znajdują się nazwy kolejek w tablicy rekordów obiektów. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (oznacza to, że niektóre kolejki zostały pomyślnie otwarte, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn). Kod przyczyny RC2136 z wywołania wskazuje tę sprawę. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna ta jest

zwracana w parametrze *REASON* wywołania MQOPEN lub MQPUT1 , a rekordy odpowiedzi nie są ustawione. Rekordy odpowiedzi są opcjonalne, ale jeśli są one podane, muszą być z nich *ODREC* .

Rekordy odpowiedzi mogą być udostępniane w taki sam sposób, jak rekordy obiektów, poprzez określenie przesunięcia w składce *ODRR0* lub przez podanie adresu w programie *ODRRP* ; zobacz opis *ODOR0* poprzednio, aby uzyskać szczegółowe informacje na temat tego, jak to zrobić. Jednak nie można użyć więcej niż jednego z produktów *ODRR0* i *ODRRP* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2156 , jeśli oba są niezerowe.

W przypadku wywołania MQPUT1 te rekordy odpowiedzi są używane do zwracania informacji o błędach, które występują, gdy komunikat jest wysyłany do kolejek na liście dystrybucyjnej, a także błędów, które występują podczas otwierania kolejek. Kod zakończenia i kod przyczyny z operacji put dla kolejki zastępują te z operacji otwarcia dla tej kolejki tylko wtedy, gdy kod zakończenia z tej operacji był kodem CCOK lub CCWARN.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODRRP (wskaźnik)

Adres pierwszego rekordu odpowiedzi.

Jest to adres pierwszego rekordu odpowiedzi MQRR. *ODRRP* jest używany tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli wartość *ODREC* wynosi zero.

ODRRP lub *ODRR0* mogą być używane do określania rekordów odpowiedzi, ale nie do obu tych wartości. Szczegółowe informacje można znaleźć w poprzednim opisie pola *ODRR0* . Jeśli produkt *ODRRP* nie jest używany, należy go ustawić na pusty wskaźnik lub zerową liczbę bajtów.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

ODSIDV

Identyfikator struktury deskryptora obiektu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest ODSIDV.

ODSS (MQCHARV)

ODSS zawiera łańcuch używany do udostępniania kryteriów wyboru używanych podczas pobierania komunikatów z kolejki.

Produkt *ODSS* nie może być udostępniany w następujących przypadkach:

- Jeśli *ODOT* nie jest OTQ
- Jeśli otwierana kolejka nie jest otwierana za pomocą jednej z opcji wejściowych, OOINP*

Jeśli w takich przypadkach zostanie podana wartość *ODSS* , wywołanie nie powiedzie się i zostanie zakodowany kod przyczyny RC2516.

Jeśli wartość *ODSS* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2519. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER4*.

ODUDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba pomyślnie otwartych kolejek zdalnych

Jest to liczba kolejek znajdujących się na liście dystrybucyjnej, które są rozstrzygane do kolejek zdalnych i które zostały pomyślnie otwarte. Jeśli ta wartość jest obecna, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż *ODVER2*.

ODVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

ODVER1

Struktura deskryptora obiektu Version-1 .

ODVER2

Struktura deskryptora obiektu Version-2 .

ODVER3

Struktura deskryptora obiektu Version-3 .

ODVER4

Struktura deskryptora obiektu Version-4 .

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

ODVERC

Bieżąca wersja struktury deskryptora obiektu.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to *ODVER1*.

Wartości początkowe

<i>Tabela 152. Początkowe wartości pól w MQOD</i>		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>ODSID</i>	<i>ODSIDV</i>	'OD--'
<i>ODVER</i>	<i>ODVER1</i>	1
<i>ODOT</i>	<i>OTQ</i>	1
<i>ODON</i>	Brak	Puste
<i>ODMN</i>	Brak	Puste
<i>ODDN</i>	Brak	'AMQ.*'
<i>ODAU</i>	Brak	Puste
<i>ODREC</i>	Brak	0
<i>ODKDC</i>	Brak	0
<i>ODUDC</i>	Brak	0
<i>ODIDC</i>	Brak	0
<i>ODORO</i>	Brak	0
<i>ODRR0</i>	Brak	0
<i>ODORP</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>ODRRP</i>	Brak	Pusty wskaźnik lub zerowe bajty
<i>ODASI</i>	<i>SINONE</i>	Wartości null
<i>ODRQN</i>	Brak	Puste

Tabela 152. Początkowe wartości pól w MQOD (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
ODRMN	Brak	Puste
ODOS	Zgodnie z definicją dla MQCHARV	Zgodnie z definicją dla MQCHARV
ODRO	Zgodnie z produktem ODOS	Zgodnie z produktem ODOS
ODSS	Brak	Puste
Uwagi: 1. Symbol – reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID          1          4    INZ('OD ')
D*
D* Structure version number
D  ODVER          5          8I 0 INZ(1)
D*
D* Object type
D  ODOT          9         12I 0 INZ(1)
D*
D* Object name
D  ODON         13          60    INZ
D*
D* Object queue manager name
D  ODMN         61         108    INZ
D*
D* Dynamic queue name
D  ODDN        109         156    INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU        157         168    INZ
D*
** Number of object records
D* present
D  ODREC        169        172I 0 INZ(0)
D*
** Number of local queues opened
D* successfully
D  ODKDC        173        176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC        177        180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC        181        184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO        185        188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO        189        192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP        193        208*   INZ(*NULL)
D*
** Address of first response
D* record
D  ODRRP        209        224*   INZ(*NULL)

```

```

D*
D* Alternate security identifier
D  ODASI                225      264      INZ(X'000000000000000000-
D                                000000000000000000000000-
D                                000000000000000000000000-
D                                000000000000')
D*
D* Resolved queue name
D  ODRQN                265      312      INZ
D*
D* Resolved queue manager name
D  ODRMN                313      360      INZ
D*
D* reserved field
D  ODRE1                361      364I 0 INZ(0)
D*
D* reserved field
D  ODRS2                365      368I 0 INZ(0)
D*
D* Object long name
D* Address of variable length string
D  ODOSCHRP            369      384*      INZ(*NULL)
D* Offset of variable length string
D  ODOSCHRO            385      388I 0 INZ(0)
D* Size of buffer
D  ODOSVSBS            389      392I 0 INZ(-1)
D* Length of variable length string
D  ODOSCHRL            393      396I 0 INZ(0)
D* CCSID of variable length string
D  ODOSCHRC            397      400I 0 INZ(-3)
D*
D* Message Selector
D* Address of variable length string
D  ODSSCHRP            401      416*      INZ(*NULL)
D* Offset of variable length string
D  ODSSCHRO            417      420I 0 INZ(0)
D* Size of buffer
D  ODSSVSBS            421      424I 0 INZ(-1)
D* Length of variable length string
D  ODSSCHRL            425      428I 0 INZ(0)
D* CCSID of variable length string
D  ODSSCHRC            429      432I 0 INZ(-3)
D*
D* Resolved long object name
D* Address of variable length string
D  ODRSOCHRP            433      448*      INZ(*NULL)
D* Offset of variable length string
D  ODRSOCHRO            449      452I 0 INZ(0)
D* Size of buffer
D  ODRSOVSBS            453      456I 0 INZ(-1)
D* Length of variable length string
D  ODRSOCHRL            457      460I 0 INZ(0)
D* CCSID of variable length string
D  ODRSOCHRC            461      464I 0 INZ(-3)
D*
D* Alias queue resolved object type
D  ODRT                465      468I 0 INZ(0)

```

MQOR-rekord obiektu

Struktura MQOR służy do określania nazwy kolejki i nazwy menedżera kolejek dla pojedynczej kolejki docelowej.

Przegląd

Cel: MQOR jest strukturą wejściową dla wywołań MQOPEN i MQPUT1 .

Zestaw znaków i kodowanie: Dane w tabeli MQOR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniu MQOPEN, możliwe jest otwarcie listy kolejek; ta lista jest nazywana *listą dystrybucyjną*. Jeśli kolejka została otwarta pomyślnie, każdy komunikat

umieszczany przy użyciu uchwytu kolejki zwróconego przez wywołanie MQOPEN jest umieszczany na każdej z kolejek na liście.

- [“Pola” na stronie 1187](#)
- [“Wartości początkowe” na stronie 1187](#)
- [“Deklaracja RPG” na stronie 1187](#)

Pola

Struktura MQOR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

ORMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek obiektów.

Jest to taka sama sytuacja, jak w przypadku pola *ODMN* w strukturze MQOD (szczegółowe informacje na ten temat zawiera tabela MQOD).

To jest zawsze pole wejściowe. Początkowa wartość tego pola to 48 znaków odstępu.

ORON (48-bajtowy łańcuch znaków)

Nazwa obiektu.

Jest to taka sama sytuacja, jak w przypadku pola *ODON* w strukturze MQOD (szczegółowe informacje na ten temat zawiera tabela MQOD), z tym wyjątkiem, że:

- Musi to być nazwa kolejki.
- Nie może to być nazwa kolejki modelowej.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to 48 znaków odstępu.

Wartości początkowe

Tabela 153. Wartości początkowe pól w tabeli MQOR		
Nazwa pola	Nazwa stałej	Wartość stałej
ORON	Brak	Puste
ORMN	Brak	Puste

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQOR Structure
D*
D* Object name
D  ORON              1      48    INZ
D* Object queue manager name
D  ORMN              49     96    INZ
```

MQPD-deskryptor właściwości

MQPD służy do definiowania atrybutów właściwości.

Przegląd

Przeznaczenie: Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSETMP i parametrem wyjściowym w wywołaniu MQINQMP.

Zestaw znaków i kodowanie: Dane w tabeli MQPD muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1188](#)
- [“Wartości początkowe” na stronie 1190](#)
- [“Deklaracja RPG” na stronie 1191](#)

Pola

Struktura MQPD zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

PDCT (10-cyfrowa liczba całkowita ze znakiem)

W tym temacie opisano kontekst komunikatu, do którego należy właściwość.

Po odebraniu przez menedżer kolejek komunikatu zawierającego właściwość zdefiniowaną przez produkt IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek koryguje wartość pola *PDCT*.

Można określić następujące opcje:

PDUSC

Właściwość jest powiązana z kontekstem użytkownika.

Do ustawienia właściwości powiązanej z kontekstem użytkownika przy użyciu wywołania MQSETMP nie jest wymagana żadna specjalna autoryzacja.

W przypadku menedżera kolejek produktu IBM WebSphere MQ 7.0 właściwość powiązana z kontekstem użytkownika jest zapisywana w sposób opisany w sekcji OOSAVA. Wywołanie MQPUT z określonym PMPASA powoduje, że właściwość zostanie skopiowana z zapisanego kontekstu do nowego komunikatu.

Jeśli wcześniej opisana opcja nie jest wymagana, można użyć następującej opcji:

PDNOC

Ta właściwość nie jest powiązana z kontekstem komunikatu.

Nierozpoznana wartość jest odrzucana przy użyciu kodu *PDREA* RC2482.

Jest to pole wejściowe/wyjściowe w wywołaniu MQSETMP i w polu wyjściowym z wywołania MQINQMP. Wartością początkową tego pola jest PDNOC.

PDCPYOPT (10-cyfrowa liczba całkowita ze znakiem)

W tym temacie opisano typy komunikatów, do których należy skopiować właściwość.

To jest pole tylko dla danych wyjściowych dla rozpoznanych właściwości zdefiniowanych przez produkt IBM MQ; IBM MQ ustawia odpowiednią wartość.

Po odebraniu przez menedżer kolejek komunikatu zawierającego właściwość zdefiniowaną przez produkt IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek koryguje wartość pola *CopyOptions*.

Można określić jedną lub więcej spośród tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodaj tej samej stałej więcej niż raz) lub połącz wartości przy użyciu operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

COPFOR

Ta właściwość jest kopiowana do przekazywanego komunikatu.

COPPUB

Ta właściwość jest kopiowana do komunikatu odebranego przez subskrybenta, gdy jest publikowany komunikat.

COPREP

Ta właściwość jest kopiowana do komunikatu odpowiedzi.

COPRP

Ta właściwość jest kopiowana do komunikatu raportu.

COPALL

Ta właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

COPNON

Ta właściwość nie jest kopiowana do komunikatu.

Opcja domyślna: W celu podania domyślnego zestawu opcji kopiowania można podać następującą opcję:

COPDEF

Ta właściwość jest kopiowana do wiadomości przesyłanej, do komunikatu raportu lub do komunikatu odebranego przez subskrybenta w momencie publikowania komunikatu.

Jest to równoznaczne z określeniem kombinacji opcji COPFOR, plus COPRP, plus COPPUB.

Jeśli żadna z opcji opisanych wcześniej nie jest wymagana, użyj następującej opcji:

COPNON

Tej wartości należy użyć, aby wskazać, że nie określono żadnych innych opcji kopiowania; programowo nie istnieje relacja między tą właściwością a kolejnymi komunikatami. Ta właściwość jest zawsze zwracana dla właściwości deskryptora komunikatu.

Jest to pole wejściowe/wyjściowe w wywołaniu MQSETMP i w polu wyjściowym z wywołania MQINQMP. Wartością początkową tego pola jest COPDEF.

PDOPT (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

PDNONE

Nie określono opcji

To jest zawsze pole wejściowe. Wartością początkową tego pola jest PDNONE.

PDSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

PSIDV

Identyfikator struktury deskryptora właściwości.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **PSIDV**.

PDSUP (10-cyfrowa liczba całkowita ze znakiem)

W tym polu opisano poziom obsługi właściwości komunikatu wymagany przez menedżer kolejek, aby komunikat zawierający tę właściwość mógł zostać umieszczony w kolejce. Dotyczy to tylko właściwości zdefiniowanych w produkcie IBM MQ; obsługa wszystkich pozostałych właściwości jest opcjonalna.

Pole jest automatycznie ustawiane na poprawną wartość, jeśli właściwość zdefiniowana przez produkt IBM MQ jest znana przez menedżer kolejek. Jeśli właściwość nie zostanie rozpoznana, przypisano wartość PDSUPO. Po odebraniu przez menedżer kolejek komunikatu zawierającego właściwość zdefiniowaną przez produkt IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek koryguje wartość pola *PDSUP*.

Podczas ustawiania właściwości zdefiniowanej w produkcie IBM MQ za pomocą wywołania MQSETMP na uchwycie komunikatu, w którym została ustawiona opcja CMNOVA, *PDSUP* staje się polem wyjściowym. Umożliwia to aplikacji umieszczanie właściwości zdefiniowanej w produkcie IBM MQ z poprawną wartością, jeśli właściwość nie jest obsługiwana przez połączony menedżer kolejek, ale w przypadku, gdy komunikat ma być przetworzony w innym menedżerze kolejek.

Wartość PDSUPO jest zawsze przypisywany do właściwości, które nie są właściwościami zdefiniowanymi w produkcie IBM MQ.

Jeśli menedżer kolejek produktu IBM WebSphere MQ 7.0, który obsługuje właściwości komunikatu, odbiera właściwość zawierającą nierozpoznaną wartość *PDSUP*, właściwość ta jest traktowana tak, jakby:

- Określono PDSUPR, jeśli którekolwiek z nierozpoznanych wartości znajdują się w PDRUM.
- Określono PDSUPL, jeśli którekolwiek z nierozpoznanych wartości znajdują się w PDAUXM
- W przeciwnym razie określono PDSUPO.

Jedna z następujących wartości jest zwracana przez wywołanie MQINQMP lub jedna z wartości, która może zostać określona podczas używania wywołania MQSETMP dla uchwytu komunikatu, w którym ustawiona jest opcja CMNOVA:

PDSUPO

Ta właściwość jest akceptowana przez menedżera kolejek nawet wtedy, gdy nie jest obsługiwana. Tę właściwość można usunąć, aby komunikat mógł przepływać do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywana do właściwości, które nie są zdefiniowane w produkcie IBM MQ.

OCZ_ZATW

Wymagana jest obsługa właściwości. Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje zdefiniowanej przez produkt IBM MQ właściwości. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2490.

PDSUPL

Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje zdefiniowanej przez IBM MQ właściwości, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2490.

Wywołanie MQPUT lub MQPUT1 powiedzie się, jeśli komunikat jest przeznaczony dla zdalnego menedżera kolejek.

Jest to pole wyjściowe w wywołaniu MQINQMP i pole wejściowe w wywołaniu MQSETMP, jeśli uchwyt komunikatu został utworzony za pomocą zestawu opcji CMNOVA. Wartością początkową tego pola jest PDSUPO.

PDVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

PDVER1

Struktura deskryptora właściwości Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

PDVERC

Bieżąca wersja struktury deskryptora właściwości.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **PDVER1**.

Wartości początkowe

Tabela 154. Początkowe wartości pól w tabeli MQPD		
Nazwa pola	Nazwa stałej	Wartość stałej
PDSID	PDSIDV	' PD '
PDVER	PDVER1	1
PDOPT	PDNONE	0
PDSUP	PDSUPO	0
PDCT	PDNOC	0
PDOPYOPT	COPDEF	0

Deklaracja RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID              1      4      INZ('DMHO')
D*
D* Structure version number
D  DMVER              5      8I 0  INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT              9      12I 0 INZ(0)
```

MQPMO-Put-message, opcje

Struktura MQPMO umożliwia aplikacji określanie opcji, które sterują sposobem umieszczania komunikatów w kolejkach lub publikowanymi w tematach.

Przegląd

Przeznaczenie

Struktura jest parametrem wejściowym/wyjściowym w wywołaniach MQPUT i MQPUT1 .

Wersja

Bieżącą wersją programu MQPMO jest PMVER2. Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach, które są następujące.

Udostępniony plik COPY zawiera najnowszą wersję programu MQPMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *PMVER* ustawioną na PMVER1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić pole *PMVER* na numer wersji wymaganej wersji.

Zestaw znaków i kodowanie

Dane w programie MQPMO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1191](#)
- [“Wartości początkowe” na stronie 1206](#)
- [“Deklaracja RPG” na stronie 1206](#)

Pola

Struktura MQPMO zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

PMCT (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt obiektu kolejki wejściowej.

Jeśli określono PMPASI lub PMPASA, to pole musi zawierać uchwyt kolejki wejściowej, z którego pobierane są informacje kontekstowe, które mają być powiązane z umieszczonym komunikatem.

Jeśli nie określono PMPASI i PMPASA, to pole jest ignorowane.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

PMIDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba wiadomości, które nie mogły zostać wysłane.

Jest to liczba komunikatów, których nie można było wysłać do kolejek znajdujących się na liście dystrybucyjnej. Liczba ta obejmuje kolejki, które nie zostały otwarte, oraz kolejki, które zostały pomyślnie otwarte, ale dla których operacja put nie powiodła się. To pole jest również

ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Uwaga: To pole jest ustawiane tylko wtedy, gdy parametr *CMPCOD* w wywołaniu MQPUT lub MQPUT1 ma wartość CCOK lub CCWARN; nie jest ustawiony, jeśli parametr *CMPCOD* ma wartość CCFAIL.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMKDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba komunikatów wystanych pomyślnie do kolejek lokalnych.

Jest to liczba komunikatów, które bieżące wywołanie MQPUT lub MQPUT1 zostało pomyślnie wysłane do kolejek na liście dystrybucyjnej, które są kolejkami lokalnymi. Liczba nie obejmuje komunikatów wysyłanych do kolejek, które są rozstrzygane do kolejek zdalnych (nawet jeśli początkowo używana jest lokalna kolejka transmisji do przechowywania komunikatu). To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje, które sterują działaniem MQPUT i MQPUT1.

Można podać dowolną z poniższych opcji lub nie można jej określić. Jeśli wymagane jest dodanie więcej niż jednego, wartości można dodać (nie należy dodawać tej samej stałej więcej niż raz). Podane kombinacje nie są poprawne; wszystkie pozostałe kombinacje są poprawne.

Opcje publikowania: Następujące opcje sterują sposobem publikowania komunikatów w temacie.

PMSRTO

Wszystkie informacje wypełnione w polach MDRQ i MDRM deskryptora MQMD tej publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja jest używana z opcją raportu, która wymaga kolejki ReplyTo, wywołanie kończy się niepowodzeniem z produktem RC2027.

PMRET

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Dzięki temu subskrybent może zażądać kopii tej publikacji po tym, kiedy został on opublikowany, za pomocą wywołania MQSUBRQ. Umożliwia również wysłanie publikacji do aplikacji, które dokonają ich subskrypcji po chwili dokonania tej publikacji, chyba że zdecydowały się nie wysyłać jej za pomocą opcji SONEWP. Jeśli aplikacja wysyła publikację, która została zachowana, jest ona wskazana przez właściwość komunikatu mq.IsRetained w tej publikacji.

Tylko jedna publikacja może być przechowywana w każdym węźle drzewa tematów. Oznacza to, że jeśli istnieje już zachowana publikacja dla tego tematu, opublikowana przez dowolną inną aplikację, zostanie ona zastąpiona tą publikacją. W związku z tym lepiej jest unikać posiadania więcej niż jednego publikatora zachowującego wiadomości na ten sam temat.

Jeśli subskrybent żąda zachowanych publikacji, użyta subskrypcja może zawierać znak wieloznaczny w temacie, w którym to przypadku może być zgodna liczba zachowanych publikacji (w różnych węzłach w drzewie tematów), a do aplikacji żądającej może być wysłanych kilka publikacji. Więcej informacji można znaleźć w opisie wywołania [“MQSUBRQ-żądanie subskrypcji” na stronie 807](#).

Jeśli ta opcja jest używana i publikacja nie może zostać zachowana, komunikat nie zostanie opublikowany, a wywołanie zakończy się niepowodzeniem z programem RC2479.

Opcje punktu synchronizacji: Następujące opcje odnoszą się do udziału wywołania MQPUT lub MQPUT1 w ramach jednostki pracy:

PMSYP

Umieść komunikat z elementem sterującym punktu synchronizacji.

Żądanie ma działać w ramach normalnych protokołów jednostkowych pracy. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

Jeśli ta opcja i PMNSYP nie zostaną określone, żądanie umieszczenia nie znajduje się w obrębie jednostki pracy.

Parametr PMSYP nie może być określony z PMNSYP.

PMNSYP

Umieść komunikat bez elementu sterującego punktu synchronizacji.

Wniosek ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest dostępny natychmiast i nie można go usunąć, tworząc kopię zapasową jednostki pracy.

Jeśli ta opcja i PMSYP nie zostaną określone, żądanie umieszczenia nie znajduje się w obrębie jednostki pracy.

Parametr PMNSYP nie może być określony w programie PMSYP.

Opcje identyfikatora komunikatu i identyfikatora korelacji: Następujące opcje żądają wygenerowania przez menedżer kolejek nowego identyfikatora komunikatu lub identyfikatora korelacji:

Identyfikator PMNMID

Wygeneruj nowy identyfikator komunikatu.

Ta opcja powoduje, że menedżer kolejek zastąpi zawartość pola *MDMID* w strukturze MQMD z nowym identyfikatorem komunikatu. Ten identyfikator komunikatu jest wysyłany razem z komunikatem i jest zwracany do aplikacji na wyjściu z wywołania MQPUT lub MQPUT1 .

Tę opcję można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej; szczegółowe informacje można znaleźć w opisie pola *PRMID* w strukturze MQPMR.

Użycie tej opcji zwalnia z zastosowania potrzeby zresetowania pola *MDMID* na MINONE przed każdym wywołaniem MQPUT lub MQPUT1 .

Identyfikator PMNCID

Wygeneruj nowy identyfikator korelacji.

Ta opcja powoduje, że menedżer kolejek zastąpi zawartość pola *MDCID* w strukturze MQMD z nowym identyfikatorem korelacji. Ten identyfikator korelacji jest wysyłany z komunikatem i zwracany do aplikacji na wyjściu z wywołania MQPUT lub MQPUT1 .

Tę opcję można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej; szczegółowe informacje można znaleźć w opisie pola *PRCID* w strukturze MQPMR.

Identyfikator PMNCID jest przydatny w sytuacjach, w których aplikacja wymaga unikalnego identyfikatora korelacji.

Opcje grupy i segmentu: Poniższa opcja odnosi się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Definicje te mogą być pomocne w zrozumieniu tej opcji:

Komunikat fizyczny

Jest to najmniejsza jednostka informacji, która może zostać umieszczona w kolejce lub usunięta z kolejki. Jest ona często zgodna z informacjami podanymi lub pobraną w pojedynczej operacji MQPUT, MQPUT1 lub MQGET. Każdy komunikat fizyczny ma własny deskryptor komunikatu (MQMD). Ogólnie, komunikaty fizyczne wyróżniają się różnymi wartościami dla identyfikatora komunikatu (pole *MDMID* w strukturze MQMD), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny

Jest to pojedyncza jednostka informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny byłby taki sam, jak komunikat fizyczny. Jednak w przypadku, gdy komunikaty logiczne są duże, ograniczenia systemowe mogą być zalecane lub konieczne, aby podzielić komunikat logiczny na dwa lub więcej komunikatów fizycznych, zwanych *segmentami*.

Komunikat logiczny, który został posegmentowany, składa się z dwóch lub większej liczby komunikatów fizycznych o tym samym identyfikatorze grupy niepustych (pole *MDGID* w strukturze MQMD) i o tym samym numerze kolejnym komunikatu (pole *MDSEQ* w strukturze MQMD). Segmenty są rozróżniane przez różne wartości dla przesunięcia segmentu (pole *MDOFF* w strukturze MQMD), co daje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dopuszczona przez aplikację wysyłającą, ma również identyfikator grupy niezerowej, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają identyfikator grupy o wartości NULL (GINONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy

Jest to zestaw jednego lub większej liczby komunikatów logicznych, które mają ten sam niepusty identyfikator grupy. Komunikaty logiczne w grupie są rozróżniane przez różne wartości dla numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do n, gdzie n jest liczbą komunikatów logicznych w grupie. Jeśli co najmniej jeden komunikat logiczny jest segmentowany, w grupie jest więcej niż n komunikatów fizycznych.

PMLOGO

Komunikaty w grupach i segmentach komunikatów logicznych są umieszczane w porządku logicznym.

Ta opcja informuje menedżera kolejek o tym, w jaki sposób aplikacja umieszcza komunikaty w grupach i segmentach komunikatów logicznych. Można ją określić tylko w wywołaniu MQPUT. Nie jest ona poprawna w wywołaniu metody MQPUT1.

Jeśli określono PMLOGO, oznacza to, że aplikacja używa kolejnych wywołań MQPUT do:

- Umieszczenie segmentów w każdym komunikacie logicznym w kolejności rosnącego przesunięcia segmentu, począwszy od 0, bez przerw.
- Umieść wszystkie segmenty w jednym komunikacie logicznym przed umieszczeniem segmentów w następnym komunikacie logicznym.
- Umieszczenie komunikatów logicznych w każdej grupie komunikatów w kolejności rosnących numerów kolejnych komunikatów, począwszy od 1, bez przerw.
- Przed umieszczeniem komunikatów logicznych w następnej grupie komunikatów należy umieścić wszystkie komunikaty logiczne w jednej grupie komunikatów.

Ta kolejność jest nazywana "porządkiem logicznym".

Ponieważ aplikacja opowiedziała menedżerowi kolejek, w jaki sposób umieszcza komunikaty w grupach i segmentach komunikatów logicznych, aplikacja nie musi obsługiwać i aktualizować informacji o grupach i segmentach na temat poszczególnych wywołań MQPUT, co powoduje, że menedżer kolejek wykonuje tę operację. W szczególności oznacza to, że aplikacja nie musi ustawiać pól *MDGID*, *MDSEQ* i *MDOFF* w strukturze MQMD, ponieważ menedżer kolejek ustawia te wartości na odpowiednie wartości. Aplikacja musi ustawić tylko pole *MDMFL* w strukturze MQMD, aby wskazać, kiedy komunikaty należą do grup lub są segmentami komunikatów logicznych, a także w celu wskazania ostatniego komunikatu w grupie lub ostatnim segmencie komunikatu logicznego.

Po uruchomieniu grupy komunikatów lub komunikatu logicznego kolejne wywołania MQPUT muszą określać odpowiednie flagi MF* w produkcie *MDMFL* w strukturze MQMD. Jeśli aplikacja próbuje umieścić komunikat nie w grupie, gdy istnieje niezakończona grupa komunikatów lub komunikat, który nie jest segmentem w przypadku wystąpienia niezakończonego komunikatu logicznego, wywołanie nie powiedzie się z kodem przyczyny RC2241 lub RC2242, w zależności od przypadku. Menedżer kolejek zachowuje jednak informacje na temat bieżącej grupy komunikatów lub bieżącego komunikatu logicznego, a aplikacja może je zakończyć, wysyłając komunikat (prawdopodobnie bez danych komunikatu aplikacji), określając odpowiednio MFLMIG lub

MFLSEG, przed ponownym wywołaniem wywołania MQPUT w celu umieszczenia komunikatu, który nie znajduje się w grupie, albo nie jest to segment.

Tabela 155 na stronie 1195 przedstawia kombinacje opcji i flag, które są poprawne, oraz wartości pól MDGID, MDSEQ i MDOFF używanych przez menedżer kolejek w każdym przypadku. Kombinacje opcji i opcji, które nie są wyświetlane w tabeli, są niepoprawne. Kolumny w tabeli mają następujące znaczenia:

PROTOKÓŁ ORD

Wskazuje, czy opcja PMLOGO jest określona w wywołaniu.

MIG

Wskazuje, czy opcja MFMIG lub MFLMIG została określona w wywołaniu.

SEG

Wskazuje, czy opcja MFSEG lub MFLSEG jest określona w wywołaniu.

SEG OK

Wskazuje, czy opcja MFSEGA jest określona w wywołaniu.

Cur grp

Wskazuje, czy bieżąca grupa komunikatów istnieje przed wywołaniem.

Komunikat dziennika Cur

Wskazuje, czy bieżący komunikat logiczny istnieje przed wywołaniem.

Pozostałe kolumny

Pokaż wartości używane przez menedżer kolejek. "Poprzedni" oznacza wartość użytą dla pola w poprzednim komunikacie dla uchwytu kolejki.

PMRLOC

Określa, że wartość PMRQN w strukturze MQPMO musi zostać zakończona z nazwą kolejki lokalnej, do której rzeczywiście zostanie wstawiony komunikat. Nazwa ResolvedQMgr jest podobnie wypełniana nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną. Zobacz OORLOQ za to, co to oznacza. Jeśli użytkownik ma uprawnienia do umieszczenia w kolejce, wówczas mają one uprawnienia wymagane do określenia tej flagi w wywołaniu MQPUT. Nie jest wymagane żadne uprawnienie specjalne.

Tabela 155. Opcje MQPUT odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych								
Opcje określone przez użytkownika				Status grupy i dziennika-komunikat przed wywołaniem		Wartości, których używa menedżer kolejek		
PROTOKÓŁ ORD	MIG	SEG	SEG OK	Cur grp	Komunikat dziennika Cur	MDGID	MDSEQ	MDOFF
Tak	Nie	Nie	Nie	Nie	Nie	GINONE	1	0
Tak	Nie	Nie	Tak	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Nie	Tak	Tak lub Nie	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Nie	Tak	Tak lub Nie	Nie	Tak	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + długość poprzedniego segmentu

Tabela 155. Opcje MQPUT odnoszące się do komunikatów w grupach i segmentach komunikatów logicznych (kontynuacja)

Opcje określone przez użytkownika				Status grupy i dziennika-komunikat przed wywołaniem		Wartości, których używa menedżer kolejek		
Tak	Tak	Tak lub Nie	Tak lub Nie	Nie	Nie	Identyfikator nowej grupy	1	0
Tak	Tak	Tak lub Nie	Tak lub Nie	Tak	Nie	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Tak lub Nie	Tak	Tak	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + długość poprzedniego segmentu
Nie	Nie	Nie	Nie	Tak lub Nie	Tak lub Nie	GINONE	1	0
Nie	Nie	Nie	Tak	Tak lub Nie	Tak lub Nie	Identyfikator nowej grupy, jeśli wartość GINONE, wartość else w polu	1	0
Nie	Nie	Tak	Tak lub Nie	Tak lub Nie	Tak lub Nie	Identyfikator nowej grupy, jeśli wartość GINONE, wartość else w polu	1	Wartość w polu
Nie	Tak	Nie	Tak lub Nie	Tak lub Nie	Tak lub Nie	Identyfikator nowej grupy, jeśli wartość GINONE, wartość else w polu	Wartość w polu	0
Nie	Tak	Tak	Tak lub Nie	Tak lub Nie	Tak lub Nie	Identyfikator nowej grupy, jeśli wartość GINONE, wartość else w polu	Wartość w polu	Wartość w polu

Uwaga:

- Program PMLOGO nie jest poprawny w wywołaniu MQPUT1 .
- W przypadku pola *MDMID* menedżer kolejek generuje nowy identyfikator komunikatu, jeśli określony jest parametr PMNMID lub MINONE, a w przeciwnym razie używa wartości w polu.
- W przypadku pola *MDCID* menedżer kolejek generuje nowy identyfikator korelacji, jeśli określony jest identyfikator PMNCID, a w przeciwnym razie używa wartości w polu.

Jeśli określono PMLOGO, menedżer kolejek wymaga, aby wszystkie komunikaty w grupie i segmentach w komunikacie logicznym były umieszczane z taką samą wartością w polu *MDPER*

w strukturze MQMD, to znaczy wszystkie muszą być trwałe lub wszystkie muszą być nietrwałe. Jeśli ten warunek nie zostanie spełniony, wywołanie MQPUT nie powiedzie się z kodem przyczyny RC2185 .

Opcja PMLOGO wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy komunikat fizyczny w grupie lub komunikat logiczny jest umieszczany w jednostce pracy, wszystkie inne komunikaty fizyczne w grupie lub komunikacie logicznym muszą zostać umieszczone w jednostce pracy, o ile ten sam uchwyt kolejki jest używany. Nie muszą one jednak być umieszczane w tej samej jednostce pracy. Pozwala to na rozdzielenie grupy komunikatów lub komunikatu logicznego składającego się z wielu komunikatów fizycznych na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy komunikat fizyczny w grupie lub komunikacie logicznym nie jest umieszczany w jednostce pracy, żaden inny komunikat fizyczny w grupie lub komunikat logiczny nie może zostać umieszczony w jednostce pracy, jeśli ten sam uchwyt kolejki jest używany.

Jeśli te warunki nie są spełnione, wywołanie MQPUT nie powiedzie się z kodem przyczyny RC2245 .

Jeśli określono PMLOGO, wartość MQMD podana w wywołaniu MQPUT nie może być mniejsza niż MDVER2. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2257 .

Jeśli wartość PMLOGO nie jest określona, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności i nie jest konieczne umieszczanie pełnych grup komunikatów ani pełnych komunikatów logicznych. Obowiązkiem aplikacji jest zapewnienie, że pola *MDGID*, *MDSEQ*, *MDOFF* i *MDMFL* mają odpowiednie wartości.

Jest to technika, której można użyć do zrestartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić pola *MDGID*, *MDSEQ*, *MDOFF*, *MDMFL* i *MDPER* na odpowiednie wartości, a następnie wywołać wywołanie MQPUT z parametrem PMSYP lub PMNSYP jako *niezbędne*, ale bez określania PMLOGO. Jeśli to wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupie i segmencie, a kolejne wywołania MQPUT używające tego uchwytu kolejki mogą określać wartość PMLOGO jako normalną.

Informacje o grupach i segmentach, które menedżer kolejek zachowuje dla wywołania MQPUT, są oddzielone od informacji o grupach i segmentach, które są zachowane dla wywołania MQGET.

W przypadku dowolnego uchwytu kolejki aplikacja może łączyć wywołania MQPUT, które określają PMLOGO z wywołaniami MQPUT, które nie są, ale należy zwrócić uwagę na następujące punkty:

- Jeśli wartość PMLOGO nie zostanie określona, każde pomyślne wywołanie MQPUT powoduje, że menedżer kolejek ustawia informacje o grupach i segmentach dla uchwytu kolejki na wartości określone przez aplikację. To zastępuje istniejące informacje o grupach i segmentach zachowane przez menedżer kolejek dla uchwytu kolejki.
- Jeśli parametr PMLOGO nie zostanie określony, wywołanie nie powiedzie się, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może jednak zakończyć się powodzeniem z kodem zakończenia CCWARN. Tabela 156 na stronie 1198 przedstawia różne przypadki, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia nie jest kodem CCOK, kod przyczyny jest jednym z następujących (odpowiednio):
 - RC2241
 - RC2242
 - RC2185
 - RC2245

Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie dla wywołania MQPUT1 .

Tabela 156. Wynik, gdy wywołanie MQPUT lub MQCLOSE nie jest spójne z informacjami o grupach i segmentach

Bieżące połączenie to	Poprzednie wywołanie było MQPUT z PMLOGO	Poprzednie wywołanie było MQPUT bez PMLOGO
MQPUT z PMLOGO	CCFAIL	CCFAIL
MQPUT bez PMLOGO	CCWARN	CCOK
MQCLOSE z niezakończonym komunikatem grupowym lub logicznym	CCWARN	CCOK

Aplikacje, które po prostu chcą umieszczać komunikaty i segmenty w porządku logicznym, są zalecane w celu określenia PMLOGO, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia aplikację z potrzeby zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tą informacją. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż podana w opcji PMLOGO, a to można osiągnąć, nie podając tej opcji. W takim przypadku aplikacja musi upewnić się, że pola *MDGID*, *MDSEQ*, *MDOFF* i *MDMFL* w strukturze MQMD są ustawione poprawnie, przed każdą wywołaniem MQPUT lub MQPUT1.

Na przykład aplikacja, która chce przekazywać wiadomości fizyczne, które otrzymuje, bez względu na to, czy te komunikaty znajdują się w grupach, czy w segmentach komunikatów logicznych, nie może określać PMLOGO. Istnieją dwa powody takiego działania:

- Jeśli komunikaty są pobierane i umieszczane w porządku, podanie parametru PMLOGO powoduje przypisanie do komunikatów nowego identyfikatora grupy, co może utrudnić lub uniemożliwić inicjatorowi komunikaty korelowanie wszystkich komunikatów odpowiedzi lub raportów, które wynikają z grupy komunikatów.
- W złożonej sieci z wieloma ścieżkami między wysyłającym i odbierającym menedżery kolejek komunikaty fizyczne mogą być odbierane poza kolejnością. Jeśli nie określono PMLOGO i odpowiadającej jej GMLOGO wywołania MQGET, aplikacja przekazująca może pobrać i przestać każdy komunikat fizyczny zaraz po nadejściu, bez konieczności oczekiwania na nadejście kolejnego komunikatu w kolejności logicznej.

Aplikacje, które generują komunikaty raportów w przypadku komunikatów w grupach lub segmentach komunikatów logicznych, nie mogą również określać PMLOGO podczas umieszczania komunikatu raportu.

PMLOGO można określić przy użyciu dowolnej z pozostałych opcji PM*.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

PMNOC

Z komunikatem nie ma być powiązany żaden kontekst.

Zarówno kontekst tożsamości, jak i kontekst źródłowy są ustawione w taki sposób, aby wskazywać brak Oznacza to, że pola kontekstu w strukturze MQMD są ustawione na:

- Odstępy dla pól znakowych
- Wartości puste dla pól typu byte
- Zera dla pól liczbowych

PMDEFC

Użyj kontekstu domyślnego.

Komunikat ma zawierać domyślne informacje o kontekście, które są z nim powiązane, zarówno dla tożsamości, jak i pochodzenia. Menedżer kolejek ustawia pola kontekstu w deskrypcorze komunikatu w następujący sposób:

Pole w strukturze MQMD	Użyta wartość
<i>MDUID</i>	Określone w środowisku, jeśli jest to możliwe; w przeciwnym razie należy ustawić odstęp.
<i>MDACC</i>	Określone ze środowiska, jeśli to możliwe; w przeciwnym razie należy ustawić wartość ACNONE.
<i>MDAID</i>	Ustaw wartość pustą.
<i>MDPAT</i>	Określone z poziomu środowiska.
<i>MDPAN</i>	Określone w środowisku, jeśli jest to możliwe; w przeciwnym razie należy ustawić odstęp.
<i>MDPD</i>	Ustaw datę, kiedy komunikat jest umieszczany.
<i>MDPT</i>	Ustaw na czas, kiedy komunikat jest umieszczany.
<i>MDAOD</i>	Ustaw wartość pustą.

Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Jest to działanie domyślne, jeśli nie zostaną określone żadne opcje kontekstu.

PMPASI

Przekaz kontekst tożsamości z uchwytu kolejki wejściowej.

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Kontekst tożsamości jest przyjmowany z uchwytu kolejki określonego w polu *PMCT*. Informacje o kontekście pochodzenia są generowane przez menedżer kolejek w taki sam sposób, jak dla PMDEFC (patrz poprzednia tabela dla wartości). Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Dla wywołania MQPUT kolejka musi zostać otwarta z opcją OOPASI (lub z opcją, która to oznacza). Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją OOPASI.

PMPASA

Przekaz cały kontekst z uchwytu kolejki wejściowej.

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Zarówno kontekst tożsamości, jak i kontekst źródłowy są pobierane z uchwytu kolejki określonego w polu *PMCT*. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Dla wywołania MQPUT kolejka musi zostać otwarta z opcją OOPASA (lub z opcją, która ją implikuje). W przypadku wywołania MQPUT1 ta sama kontrola autoryzacji jest przeprowadzana w taki sam sposób, jak w wywołaniu MQOPEN z opcją OOPASA.

PMSETI

Ustaw kontekst tożsamości z aplikacji.

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Aplikacja określa kontekst tożsamości w strukturze MQMD. Informacje o kontekście pochodzenia są generowane przez menedżer kolejek w taki sam sposób, jak dla PMDEFC (patrz poprzednia tabela dla wartości). Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Dla wywołania MQPUT kolejka musi zostać otwarta z opcją OOSSETI (lub z opcją, która to oznacza). Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją OOSSETI.

PMSETA

Ustaw cały kontekst z aplikacji.

Komunikat ma zawierać powiązane z nim informacje kontekstowe. Aplikacja określa kontekst tożsamości i pochodzenia w strukturze MQMD. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

W przypadku wywołania MQPUT kolejka musi zostać otwarta z opcją OOSETA. Dla wywołania MQPUT1 jest to ta sama kontrola autoryzacji, która jest przeprowadzana dla wywołania MQOPEN z opcją OOSETA.

Można określić tylko jedną z opcji kontekstu PM*. Jeśli żadna z tych opcji nie zostanie określona, przyjmowany jest PMDEFC.

Typy odpowiedzi umieszczania. Następujące opcje kontrolują odpowiedź zwróconej do wywołania MQPUT lub MQPUT1. Można określić tylko jedną z tych opcji. Jeśli nie określono wartości PMARES i PMSRES, przyjmuje się, że PMRASQ lub PMRAST.

PMARES

Opcja PMARES żąda, aby operacja MQPUT lub MQPUT1 została zakończona bez oczekiwania aplikacji na zakończenie wywołania przez menedżer kolejek. Użycie tej opcji może zwiększyć wydajność przesyłania komunikatów, szczególnie w przypadku aplikacji korzystających z powiązań klienta. Aplikacja może okresowo sprawdzać, używając komendy MQSTAT, niezależnie od tego, czy wystąpił błąd podczas poprzednich wywołań asynchronicznych.

W przypadku tej opcji gwarantowane jest zakończenie tylko następujących pól w strukturze MQMD;

- MDAID
- MDPAT
- MDPAN
- MDAOD

Dodatkowo, jeśli jako opcje zostaną określone wartości PMNMID lub PMNCID, zwrócone zostaną również zwrócone identyfikatory MDMID i MDCID. (Identyfikator PMNMID można jawnie określić, podając puste pole MDMID).

Tylko określone wcześniej pola zostaną zakończone. Inne informacje, które normalnie zostaną zwrócone w strukturze MQMD lub MQPMO, nie są zdefiniowane.

Podczas żądania asynchronicznej odpowiedzi put dla operacji MQPUT lub MQPUT1, parametr CMPCOD i PRZYCZYNA CCOK i RCNONE nie musi oznaczać, że komunikat został pomyślnie umieszczony w kolejce. Podczas tworzenia aplikacji MQI, która używa asynchronicznej odpowiedzi put, i wymaga potwierdzenia, że komunikaty zostały umieszczone w kolejce, należy sprawdzić zarówno kody CMPCOD, jak i REASON z operacji put, a także użyć komendy MQSTAT w celu wysłania zapytania o asynchroniczne informacje o błędach.

Mimo że powodzenie lub niepowodzenie każdego pojedynczego wywołania MQPUT/MQPUT1 może nie zostać zwrócone natychmiast, pierwszy błąd, który wystąpił w wywołaniu asynchronicznym, można określić w późniejszym momencie przy użyciu wywołania MQSTAT.

Jeśli komunikat trwały w punkcie synchronizacji nie zostanie dostarczony przy użyciu asynchronicznej odpowiedzi put, a użytkownik podejmie próbę zatwierdzenia transakcji, zatwierdzenie nie powiedzie się, a transakcja zostanie wycofana z kodu zakończenia CCFAIL i z przyczyną RC2003. Aplikacja może wywołać wywołanie MQSTAT w celu określenia przyczyny niepowodzenia poprzedniej operacji MQPUT lub MQPUT1.

PMSRES

Określenie tej wartości dla opcji put w strukturze MQPMO zapewnia, że operacja MQPUT lub MQPUT1 jest zawsze emitowana synchronicznie. Jeśli operacja zakończy się pomyślnie, wszystkie pola w strukturze MQMD i MQPMO zostaną zakończone. Jest ona udostępniana w celu zapewnienia synchronicznej odpowiedzi bez względu na wartość domyślnej odpowiedzi umieszczonej w obiekcie kolejki lub tematu.

PMRASQ

Jeśli wartość ta jest określona dla wywołania MQPUT, użyty typ odpowiedzi jest przyjmowany z wartości DEFPRESP określonej w kolejce, gdy została ona otwarta przez aplikację. Jeśli aplikacja

kliencka jest połączona z menedżerem kolejek na poziomie wcześniejszym niż Wersja 7.0, zachowuje się tak, jakby była określona wartość PMSRES.

Jeśli ta opcja jest określona dla wywołania MQPUT1, wartość DEFPRESP z definicji kolejki nie jest używana. Jeśli wywołanie MQPUT1 używa PMSYP, zachowuje się on jak dla PMARES, a jeśli używa PMNSYP, zachowuje się on jak w przypadku PMSRES.

PMRAST

Jest to synonim PMRASQ w celu użycia z obiektami tematu.

Inne opcje: Następujące opcje kontrolują sprawdzanie autoryzacji i to, co się dzieje, gdy menedżer kolejek jest wygaszany:

PMALTU

Sprawdź poprawność z określonym identyfikatorem użytkownika.

Oznacza to, że pole *ODAU* w parametrze *OBJDSC* wywołania MQPUT1 zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności uprawnień do umieszczania komunikatów w kolejce. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy *ODAU* jest autoryzowany do otwarcia kolejki z określonymi opcjami, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, jest do tego uprawniony. (Nie dotyczy to jednak określonych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym aplikacja jest uruchomiona).

Ta opcja jest poprawna tylko w przypadku wywołania MQPUT1.

PMFIQ

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

Ta opcja wymusza niepowodzenie wywołania MQPUT lub MQPUT1, jeśli menedżer kolejek znajduje się w stanie wygaszania.

Wywołanie zwraca kod zakończenia CCFAIL z kodem przyczyny RC2161.

Opcja domyślna: Jeśli żadna z opcji opisanych wcześniej nie jest wymagana, można użyć następującej opcji:

PMNONE

Nie określono żadnych opcji.

Ta wartość może być używana do wskazania, że nie zostały określone żadne inne opcje. Wszystkie opcje przyjmują wartości domyślne. PMNONE jest zdefiniowana w dokumentacji programu pomocowego; nie jest przeznaczona, aby ta opcja była używana z innymi, ale jako że jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

To jest pole wejściowe. Wartością początkową pola *PMOPT* jest PMNONE.

PMPRF (10-cyfrowa liczba całkowita ze znakiem)

Flagi wskazujące, które pola MQPMR są obecne.

To pole zawiera flagi, które muszą być ustawione w celu wskazania, które pola MQPMR są obecne w rekordach umieszczania komunikatów udostępnianych przez aplikację. *PMPRF* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero, a zarówno *PMPRO*, jak i *PMPRP* są równe zero.

W przypadku pól, które są obecne, menedżer kolejek używa dla każdego miejsca docelowego wartości z pól w odpowiednim rekordzie komunikatu umieszczonego. W przypadku pól, które są nieobecne, menedżer kolejek używa wartości ze struktury MQMD.

Można określić co najmniej jedną z następujących opcji, aby wskazać, które pola są obecne w rekordach umieszczania komunikatów:

ID_PFMID

Pole identyfikatora komunikatu jest obecne.

PFCID

Pole identyfikatora korelacji jest obecne.

Identyfikator PFGID

Pole identyfikatora grupy jest obecne.

PFFB

Pole informacji zwrotnej jest obecne.

PFACC

Pole tokenu rozliczania jest obecne.

Jeśli ta opcja jest określona, w polu *PMOPT* należy określić wartość *PMSETI* lub *PMSETA*; jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2158.

Jeśli nie ma żadnych pól *MQPMR*, można określić następujące elementy:

PFNONE

Nie istnieją pola rekordu komunikatu umieszczonego w komunikacie.

Jeśli ta wartość jest określona, wartość *PMREC* musi być zerowa albo obie wartości *PMPRO* i *PMPRP* muszą mieć wartość zero.

PFNONE jest zdefiniowane w dokumentacji programu pomocy. Nie jest zamierzone, aby ta stała była używana z innymi, ale ponieważ jej wartość jest równa zero, tego rodzaju użycie nie może zostać wykryte.

Jeśli program *PMPRF* zawiera flagi, które nie są poprawne, lub jeśli udostępnione są rekordy komunikatów, ale parametr *PMPRF* ma wartość *PFNONE*, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2158.

To jest pole wejściowe. Wartością początkową tego pola jest *PFNONE*. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMPRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu umieszczenia komunikatu od początku wywołania *MQPMO*.

Jest to przesunięcie w bajtach pierwszego rekordu komunikatu umieszczonego w *MQPMR* od początku struktury *MQPMO*. Przesunięcie może być dodatnie lub ujemne. *PMPRO* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *PMREC* wynosi zero.

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, w celu określenia określonych właściwości komunikatu dla każdego miejsca docelowego można podać tablicę jednego lub większej liczby rekordów komunikatów umieszczonych w tabeli *MQPMR*. Właściwości te są następujące:

- Identyfikator komunikatu
- identyfikator korelacji
- Identyfikator grupy
- wartość sprzężenia zwrotnego
- Token rozliczania

Nie jest konieczne określanie wszystkich tych właściwości, ale niezależnie od wybranego podzbioru, pola muszą być określone we właściwej kolejności. Szczegółowe informacje można znaleźć w opisie struktury *MQPMR*.

Zwykle istnieje wiele rekordów umieszczania komunikatów, ponieważ istnieją rekordy obiektów określone przez *MQOD*, gdy lista dystrybucyjna jest otwarta; każdy rekord umieszczania komunikatów dostarcza właściwości komunikatu dla kolejki identyfikowanej przez odpowiedni rekord obiektu. Kolejki znajdujące się na liście dystrybucyjnej, które nie otwierają się, muszą nadal umieszczać dla nich rekordy komunikatów na odpowiednich pozycjach w tablicy, chociaż w tym przypadku właściwości komunikatu są ignorowane.

Liczba rekordów komunikatów umieszczonych w rekordach może być różna od liczby rekordów obiektów. Jeśli liczba rekordów umieszczania komunikatów jest mniejsza niż rekordy obiektów, to właściwości komunikatu dla miejsc docelowych, które nie zawierają rekordów komunikatów, są pobierane z odpowiednich pól w deskrypcji komunikatu *MQMD*. Jeśli rekordy komunikatów są

umieszczane w większej ilości niż rekordy obiektów, nadmiarowe rekordy nie są używane (mimo że nadal musi istnieć możliwość ich uzyskania). Rekordy umieszczania komunikatów są opcjonalne, ale jeśli są one podane, muszą być z nich *PMREC*.

Rekordy umieszczania komunikatów mogą być udostępniane w podobny sposób jak rekordy obiektów w MQOD, poprzez określenie przesunięcia w składce *PMPRO* lub przez podanie adresu w programie *PMPRP*; aby uzyskać szczegółowe informacje na temat tego, jak to zrobić, należy zapoznać się z polem *ODORO* opisanym w sekcji “MQOD-deskryptor obiektu” na stronie 1176.

Nie można użyć więcej niż jednego z produktów *PMPRO* i *PMPRP*; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2159, jeśli oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMPRP (wskaźnik)

Adres pierwszego rekordu umieszczenia komunikatu.

Jest to adres pierwszego rekordu komunikatu umieszczonego w tabeli MQPMR. *PMPRP* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *PMREC* wynosi zero.

Można użyć opcji *PMPRP* lub *PMPRO* w celu określenia rekordów umieszczania komunikatów, ale nie obu tych rekordów. Szczegółowe informacje można znaleźć w opisie pola *PMRRO*. Jeśli produkt *PMPRP* nie jest używany, należy go ustawić na pusty wskaźnik lub zerową liczbę bajtów.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMREC (10-cyfrowa liczba całkowita ze znakiem)

Liczba rekordów umieszczania komunikatów lub rekordów odpowiedzi.

Jest to liczba rekordów komunikatów umieszczonych w tabeli MQPMR lub rekordów odpowiedzi MQRR, które zostały udostępnione przez aplikację. Liczba ta może być większa od zera tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Rekordy komunikatów i rekordy odpowiedzi są opcjonalne-aplikacja nie musi udostępniać żadnych rekordów lub może wybrać opcję udostępnienia rekordów tylko jednego typu. Jeśli jednak aplikacja udostępnia rekordy obu typów, musi ona udostępniać rekordy *PMREC* dla każdego typu.

Wartość *PMREC* nie musi być taka sama, jak liczba miejsc docelowych na liście dystrybucyjnej. Jeśli udostępniono zbyt wiele rekordów, przekroczenie tej wartości nie jest używane. Jeśli podano zbyt małą liczbę rekordów, dla właściwości komunikatu dla tych miejsc docelowych, które nie mają rekordów umieszczania komunikatów, używane są wartości domyślne (patrz *PMPRO* w dalszej części tego tematu).

Jeśli wartość *PMREC* jest mniejsza od zera lub jest większa od zera, ale komunikat nie jest umieszczany na liście dystrybucyjnej, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2154.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMRMN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa docelowego menedżera kolejek.

Jest to nazwa docelowego menedżera kolejek po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *PMRQN*, i może być nazwą lokalnego menedżera kolejek.

Jeśli *PMRQN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, *PMRMN* jest nazwą grupy współużytkowania kolejki. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, *PMRQN* może być nazwą grupy współużytkowania kolejki lub nazwą menedżera kolejek, który jest elementem grupy

współużytkowania kolejki (rodzaj zwróconej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

To jest pole wyjściowe. Długość tego pola jest podana przez LNQM. Początkowa wartość tego pola to 48 znaków odstępu.

PMRQN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa kolejki docelowej.

Jest to nazwa kolejki docelowej po translacji nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa kolejki, która istnieje w menedżerze kolejek identyfikowanego przez produkt *PMRMN*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

To jest pole wyjściowe. Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

PMRRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu odpowiedzi od początku wywołania MQPMO.

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. *PMRRO* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *PMREC* wynosi zero.

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, można podać tablicę jednego lub większej liczby rekordów odpowiedzi MQRR, aby zidentyfikować kolejki, do których komunikat nie został pomyślnie wysłany (pole *RRCC* w tabeli MQRR), oraz przyczynę każdego niepowodzenia (pole *RRREA* w tabeli MQRR). Być może komunikat nie został wysłany, ponieważ kolejka nie została otwarta lub operacja put nie powiodła się. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (oznacza to, że niektóre komunikaty zostały wysłane pomyślnie, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn); kod przyczyny RC2136 z wywołania wskazuje tę sprawę. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest zwracana w parametrze *REASON* wywołania MQPUT lub MQPUT1, a rekordy odpowiedzi nie są ustawione.

Zwykle istnieje wiele rekordów odpowiedzi, ponieważ istnieją rekordy obiektów określone przez MQOD, gdy lista dystrybucyjna jest otwierana; w razie potrzeby każdy rekord odpowiedzi jest ustawiany na kod zakończenia i kod przyczyny dla umieszczenia w kolejce identyfikowanej przez odpowiedni rekord obiektu. Kolejki na liście dystrybucyjnej, które nie otwierają się, muszą nadal mieć przypisane rekordy odpowiedzi dla odpowiednich pozycji w tablicy, chociaż są one ustawione na kod zakończenia i kod przyczyny wynikający z operacji otwarcia, a nie operacji put.

Liczba rekordów odpowiedzi jest możliwa w zależności od liczby rekordów obiektów. Jeśli liczba rekordów odpowiedzi jest mniejsza niż rekordy obiektów, aplikacja może nie być w stanie zidentyfikować wszystkich miejsc docelowych, dla których operacja put nie powiodła się, lub przyczyn niepowodzeń. Jeśli istnieje więcej rekordów odpowiedzi niż rekordy obiektów, nadwyżka nie jest używana (choć nadal musi istnieć możliwość uzyskania dostępu do nich). Rekordy odpowiedzi są opcjonalne, ale jeśli są one podane, muszą być z nich *PMREC*.

Rekordy odpowiedzi można dostarczyć w podobny sposób do rekordów obiektów w MQOD, podając przesunięcie w składce *PMRRO* lub podając adres w programie *PMRRP*. aby uzyskać szczegółowe informacje na temat tego, jak to zrobić, należy zapoznać się z polem *ODORO* opisanym w sekcji "MQOD-deskryptor obiektu" na stronie 1176. Jednak nie można użyć więcej niż jednego z produktów *PMRRO* i *PMRRP*; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2156, jeśli oba są niezerowe.

W przypadku wywołania MQPUT1 pole to musi być równe zero. Dzieje się tak dlatego, że informacje o odpowiedzi (jeśli są wymagane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

To jest pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMRRP (wskaźnik)

Adres pierwszego rekordu odpowiedzi.

Jest to adres pierwszego rekordu odpowiedzi *MQRR*. *PMRRP* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli wartość *PMREC* wynosi zero.

PMRRP lub *PMRRO* mogą być używane do określania rekordów odpowiedzi, ale nie do obu tych rekordów; szczegółowe informacje można znaleźć w opisie pola *PMRRO*. Jeśli produkt *PMRRP* nie jest używany, należy go ustawić na pusty wskaźnik lub zerową liczbę bajtów.

W przypadku wywołania *MQPUT1* pole to musi być pustym wskaźnikiem lub bajtami o wartości NULL. Dzieje się tak dlatego, że informacje o odpowiedzi (jeśli są wymagane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu *MQOD*.

To jest pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

PMSIDV

Identyfikator struktury opcji put-message.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest *PMSIDV*.

PMSL (MQLONG)

Poziom subskrypcji docelowej tej publikacji.

Ta publikacja otrzymuje tylko te subskrypcje o najwyższym *PMSL* mniejszym lub równym tej wartości. Wartość ta musi należeć do zakresu od zera do 9; zero oznacza najniższy poziom.

Wartością początkową tego pola jest 9.

PMTO (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest znacząca. Wartością początkową tego pola jest -1.

PMUDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba komunikatów wysłanych pomyślnie do kolejek zdalnych.

Jest to liczba komunikatów, które bieżące wywołanie *MQPUT* lub *MQPUT1* zostało pomyślnie wysłane do kolejek na liście dystrybucyjnej, które są rozstrzygane do kolejek zdalnych. Komunikaty, które menedżer kolejek zachowuje tymczasowo w formie listy dystrybucyjnej, są liczone jako liczba pojedynczych miejsc docelowych, które zawierają te listy dystrybucyjne. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

To jest pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawione, jeśli wartość *PMVER* jest mniejsza niż *PMVER2*.

PMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

PMVER1

Struktura opcji komendy put-message w wersji Version-1.

PMVER2

Struktura opcji komendy put-message w wersji Version-2 .

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

PMVERC

Bieżąca wersja struktury opcji put-message.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to PMVER1.

Wartości początkowe

Tabela 157. Początkowe wartości pól w MQPMO		
Nazwa pola	Nazwa stałej	Wartość stałej
PMSID	PMSIDV	'PMO↯'
PMVER	PMVER1	1
PMOPT	PMNONE	0
PMT0	Brak	-1
PMCT	Brak	0
PMKDC	Brak	0
PMUDC	Brak	0
PMIDC	Brak	0
PMRQN	Brak	Puste
PMRMN	Brak	Puste
PMREC	Brak	0
PMPRF	PFNONE	0
PMPRO	Brak	0
PMRRO	Brak	0
PMPRP	Brak	Pusty wskaźnik lub zerowe bajty
PMRRP	Brak	Pusty wskaźnik lub zerowe bajty
Uwaga:		
1. Symbol ↯ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQPMO Structure
D*
D* Structure identifier
D PMSID 1 4 INZ('PMO ')
D* Structure version number
D PMVER 5 8I 0 INZ(1)
D* Options that control the action of MQPUT and MQPUT1
D PMOPT 9 12I 0 INZ(0)
D* Reserved
D PMT0 13 16I 0 INZ(-1)
D* Object handle of input queue
```

```

D PMCT 17 20I 0 INZ(0)
D* Number of messages sent successfully to local queues
D PMKDC 21 24I 0 INZ(0)
D* Number of messages sent successfully to remote queues
D PMUDC 25 28I 0 INZ(0)
D* Number of messages that could not be sent
D PMIDC 29 32I 0 INZ(0)
D* Resolved name of destination queue
D PMRQN 33 80 INZ
D* Resolved name of destination queue manager
D PMRMN 81 128 INZ
D* Number of put message records or response records present
D PMREC 129 132I 0 INZ(0)
D* Flags indicating which MQPMR fields are present
D PMPRF 133 136I 0 INZ(0)
D* Offset of first put message record from start of MQPMO
D PMPRO 137 140I 0 INZ(0)
D* Offset of first response record from start of MQPMO
D PMRRO 141 144I 0 INZ(0)
D* Address of first put message record
D PMPRP 145 160* INZ(*NULL)
D* Address of first response record
D PMRRP 161 176* INZ(*NULL)
D* Original message handle
D PMOMH 177 184I 0
D* New message handle
D PMNMH 185 190I 0
D* The action being performed
D PMACT 191 194I 0
D* Reserved
D PMRE1 195 198I 0

```

MQPMR-Put-rekord komunikatu

Struktura MQPMR służy do określania różnych właściwości komunikatu dla pojedynczego miejsca docelowego, gdy komunikat jest umieszczany na liście dystrybucyjnej.

Przegląd

Cel: MQPMR jest strukturą wejścia/wyjścia dla wywołań MQPUT i MQPUT1.

Zestaw znaków i kodowanie: Dane w tabeli MQPMR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniu MQPUT lub MQPUT1, możliwe jest określenie różnych wartości dla każdej kolejki docelowej na liście dystrybucyjnej. Niektóre pola są tylko danymi wejściowymi, inne są wejścia/wyjścia.

Uwaga: Ta struktura jest nietypowa w tym, że nie ma ustalonego układu. Pola w tej strukturze są opcjonalne, a obecność lub nieobecność każdego pola jest wskazywana przez flagi w polu *PMPRF* w MQPMO. Pola, które są obecne w polu **muszą występować w następującej kolejności** :

- *PRMID*
- *PRCID*
- *PRGID*
- *PRFB*
- *PRACC*

Nieobecne pola nie zajmują miejsca w rekordzie.

Ponieważ tabela MQPMR nie ma stałego układu, definicja nie jest dostępna w pliku COPY. Programista aplikacji powinien utworzyć deklarację zawierającą pola wymagane przez aplikację, a następnie ustawić flagi w programie *PMPRF*, aby wskazać obecne pola.

- [“Pola” na stronie 1208](#)
- [“Wartości początkowe” na stronie 1209](#)

- [“Deklaracja RPG” na stronie 1209](#)

Pola

Struktura MQPMR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

PRACC (32-bajtowy łańcuch bitowy)

Token rozliczania.

Jest to znacznik rozliczeniowy, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w ten sam sposób, co pole *MDACC* w strukturze MQMD dla umieszczenia w jednej kolejce. Informacje na temat zawartości tego pola można znaleźć w opisie produktu *MDACC* w produkcie [“MQMD-deskryptor komunikatu” na stronie 1124](#) .

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

To jest pole wejściowe.

PRCID (24-bajtowy łańcuch bitowy)

Identyfikator korelacji.

Jest to identyfikator korelacji, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w ten sam sposób, co pole *MDCID* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowe, wartość w strukturze MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRCID* .

Jeśli określony jest identyfikator PMNCID, *pojedynczy* nowy identyfikator korelacji jest generowany i używany dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania identyfikatora PMNMID (patrz pole *PRMID*).

Jest to pole wejściowe/wyjściowe.

PRFB (10-cyfrowa liczba całkowita ze znakiem)

Informacja zwrotna lub kod przyczyny.

Jest to kod informacji zwrotnej, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w ten sam sposób, co pole *MDFB* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

To jest pole wejściowe.

PRGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Jest to identyfikator grupy, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w ten sam sposób, co pole *MDGID* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowe, wartość w strukturze MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRGID* . Wartość jest przetwarzana w sposób opisany w sekcji [Tabela 155 na stronie 1195](#), ale z następującymi różnicami:

- W tych przypadkach, w których zostanie użyty nowy identyfikator grupy, menedżer kolejek generuje inny identyfikator grupy dla każdego miejsca docelowego (to znaczy nie ma dwóch miejsc docelowych o tym samym identyfikatorze grupy).
- W przypadkach, w których wartość w polu będzie używana, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2258.

Jest to pole wejściowe/wyjściowe.

PRMID (24-bajtowy łańcuch bitowy)

Identyfikator komunikatu.

Jest to identyfikator komunikatu, który ma być używany dla komunikatu wysłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1. Jest on przetwarzany w ten sam sposób, co pole *MDMID* w strukturze MQMD dla umieszczenia w jednej kolejce.

Jeśli to pole nie jest obecne w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowe, wartość w strukturze MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRMID*. Jeśli ta wartość to MINONE, dla *każdego* z tych miejsc docelowych zostanie wygenerowany nowy identyfikator komunikatu (to znaczy, że żadne dwa z tych miejsc docelowych nie mają takiego samego identyfikatora komunikatu).

Jeśli określono parametr PMNMID, nowe identyfikatory komunikatów są generowane dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania identyfikatora PMNCID (patrz pole *PRCID*).

Jest to pole wejściowe/wyjściowe.

Wartości początkowe

Dla tej struktury nie zdefiniowano wartości początkowych, ponieważ nie podano deklaracji struktury. W poniższej przykładowej deklaracji pokazano, w jaki sposób struktura powinna być zadeklarowana przez programistę aplikacji, jeśli wszystkie pola są wymagane.

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQPMR Structure
D*
D* Message identifier
D  PRMID                1      24
D* Correlation identifier
D  PRCID                25     48
D* Group identifier
D  PRGID                49     72
D* Feedback or reason code
D  PRFB                73     76I 0
D* Accounting token
D  PRACC               77     108
```

MQRFH-nagłówek reguł i formatowania

Struktura MQRFH definiuje układ reguł i nagłówek formatowania.

Przegląd

Przeznaczenie: Ten nagłówek może być używany do wysyłania danych łańcuchowych w postaci par nazwa/wartość.

Nazwa formatu: FMRFH.

Zestaw znaków i kodowanie: pola w strukturze MQRFH (w tym *RFNVS*) znajdują się w zestawie znaków i kodowaniu podanym w polach *MDCSI* i *MDENC* w strukturze nagłówka poprzedzającym MQRFH lub przez te pola w strukturze MQMD, jeśli wartość MQRFH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

- [“Pola” na stronie 1210](#)
- [“Wartości początkowe” na stronie 1212](#)
- [“Deklaracja RPG” na stronie 1212](#)

Pola

Struktura MQRFH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

RFCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków dla danych, które są następujące *RFNVS*.

Określa identyfikator zestawu znaków dla danych, które są następujące *RFNVS* ; nie ma zastosowania do danych znakowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest *ATBRKR*.

Wartością początkową tego pola jest CSUNDF.

Kodowanie numeryczne danych, które są następujące *RFNVS*.

Określa kodowanie liczbowe dla danych, które są następujące *RFNVS* ; nie ma zastosowania do danych liczbowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest ENNAT.

RFFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Można określić następujące elementy:

RFNONE

Brak flag.

Wartością początkową tego pola jest RFNONE.

RFFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych, które są następujące *RFNVS*.

Określa nazwę formatu danych, które są następujące *RFNVS*.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

RFLN (10-cyfrowa liczba całkowita ze znakiem)

Łączna długość MQRFH, w tym *RFNVS*.

Jest to długość (w bajtach) struktury MQRFH, w tym pole *RFNVS* na końcu struktury. Długość *nie* obejmuje żadnych danych użytkownika, które są następujące po polu *RFNVS*.

Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, należy rozważyć użycie *RFLN* jako wielokrotnej liczby czterech.

Następująca stała określa długość części *stałej* struktury, to znaczy długość, z wyłączeniem pola *RFNVS*:

RFLENV

Długość stałej części struktury MQRFH.

Wartością początkową tego pola jest RFLENV.

RFNVS (łańcuch znaków n-byte)

Łańcuch zawierający pary nazwa/wartość.

Jest to łańcuch znaków o zmiennej długości zawierający pary nazwa/wartość w postaci:

```
name1 value1 name2 value2 name3 value3 ...
```

Każda nazwa lub wartość musi być oddzielona od przylegającej nazwy lub wartości przez jeden lub więcej znaków odstępu; te odstępy nie są znaczące. Nazwa lub wartość może zawierać spacje, poprzedzając je przedrostkiem i przyrostem nazwy lub wartości znakiem cudzysłowu; wszystkie znaki między otwierającym znakiem cudzysłowu a pasującym znakiem cudzysłowu zamykającym są traktowane jako znaczące. W poniższym przykładzie nazwą jest FAMOUS_WORDS, a wartością jest Hello World:

```
FAMOUS_WORDS "Hello World"
```

Nazwa lub wartość może zawierać dowolne znaki inne niż znak o kodzie zero (który działa jako ogranicznik dla produktu *RFNVS*). Jednak w celu ułatwienia współdzielenia aplikacja może preferować ograniczanie nazw do następujących znaków:

- Pierwszy znak: wielkie lub małe litery (od A do Z, od a do z) lub podkreślenie.
- Kolejne znaki: wielkie lub małe litery, cyfry dziesiętne (od 0 do 9), podkreślenie, myślnik lub kropka.

Jeśli nazwa lub wartość zawiera jeden lub więcej cudzysłowów, nazwa lub wartość muszą być ujęte w znaki cudzysłowu, a każdy znak cudzysłowu w łańcuchu musi być podwojony:

```
Famous_Words "The program displayed ""Hello World"""
```

W nazwach i wartościach rozróżniana jest wielkość liter, oznacza to, że małe litery nie są traktowane tak samo, jak wielkie litery. Na przykład: FAMOUS_WORDS i Famous_Words to dwie różne nazwy.

Długość w bajtach *RFNVS* jest równa *RFLN* minus RFLENV. Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, zaleca się, aby ta długość była wielokrotnością liczby czterech. Wartość *RFNVS* musi być dopełniona odstępami do tej długości lub przzerwana wcześniej, umieszczając znak o kodzie zero po ostatnim znaczącym znaku w łańcuchu. Znak o kodzie zero i bajty następujące po nim, aż do określonej długości *RFNVS*, są ignorowane.

Uwaga: Ponieważ długość tego pola nie jest ustalona, to pole jest pomijane w deklaracjach struktury, które są udostępniane dla obsługiwanych języków programowania.

RFSID (łańcuch znakowy 4-bajtowy)

Identyfikator struktury.

Wartość musi być następująca:

RFSIDV

Identyfikator reguł i struktury nagłówka formatowania.

Wartością początkową tego pola jest RFSIDV.

RFVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RFVER1

Reguły Version-1 i struktura nagłówka formatowania.

Początkowa wartość tego pola to RFVER1.

Wartości początkowe

Tabela 158. Początkowe wartości pól w MQRFH		
Nazwa pola	Nazwa stałej	Wartość stałej
RFSID	RFSIDV	'RFH¬'
RFVER	RFVER1	1
RFLen	RFLenV	32
RFENC	ENNAT	Zależy od środowiska
RFCSI	CSUNDF	0
RFFMT	FMNONE	Puste
RFFLG	RFNONE	0
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQRFH Structure
D*
D* Structure identifier
D  RFSID          1      4      INZ('RFH ')
D* Structure version number
D  RFVER          5      8I 0 INZ(1)
D* Total length of MQRFH includingNameValueString
D  RFLen          9      12I 0 INZ(32)
D* Numeric encoding of data that followsNameValueString
D  RFENC          13     16I 0 INZ(273)
D* Character set identifier of data thatfollows NameValueString
D  RFCSI          17     20I 0 INZ(0)
D* Format name of data that followsNameValueString
D  RFFMT          21     28      INZ(' ')
D* Flags
D  RFFLG          29     32I 0 INZ(0)
```

MQRFH2 -reguły i nagłówek formatowania 2

Struktura MQRFH2 definiuje format reguł i nagłówków formatowania version-2 .

Przegląd

Przeznaczenie: Ten nagłówek może być używany do wysyłania danych, które zostały zakodowane przy użyciu składni typu XML. Komunikat może zawierać dwie lub więcej struktur MQRFH2 z serii, a dane użytkownika są opcjonalnie następujące po ostatniej strukturze MQRFH2 w serii.

Nazwa formatu: FMRFH2.

Zestaw znaków i kodowanie: reguły specjalne mają zastosowanie do zestawu znaków i kodowania używanego w strukturze MQRFH2 :

- Pola inne niż *RF2NVD* znajdują się w zestawie znaków i kodowaniu podanym w polach *MDCSI* i *MDENC* w strukturze nagłówka, które poprzedzają MQRFH2, lub przez te pola w strukturze MQMD, jeśli MQRFH2 znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Jeśli w wywołaniu MQGET określono wartość GMCONV, to menedżer kolejek przekształca te pola w żądany zestaw znaków i kodowanie.

- *RF2NVD* znajduje się w zestawie znaków podanym w polu *RF2NVC* . Tylko niektóre zestawy znaków Unicode są poprawne dla *RF2NVC* (szczegółowe informacje można znaleźć w opisie produktu *RF2NVC*).

Niektóre zestawy znaków mają reprezentację, która jest zależna od kodowania. Jeśli *RF2NVC* jest jednym z tych zestawów znaków, produkt *RF2NVD* musi znajdować się w tym samym kodowaniu, co inne pola w tabeli MQRFH2.

Jeśli w wywołaniu MQGET określono wartość GMCONV, menedżer kolejek przekształca produkt *RF2NVD* na żądane kodowanie, ale nie zmienia jego zestawu znaków.

- [“Pola” na stronie 1213](#)
- [“Wartości początkowe” na stronie 1218](#)
- [“Deklaracja RPG” na stronie 1218](#)

Pola

Struktura MQRFH2 zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

RF2CSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych, który jest zgodny z ostatnim polem *RF2NVD* .

Określa identyfikator zestawu znaków dla danych, które są następujące po ostatnim polu *RF2NVD* . Nie ma on zastosowania do danych znakowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest ATBRKR.

Wartością początkową tego pola jest CSINHT.

RF2ENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych, które następują po ostatnim polu *RF2NVD* .

Określa kodowanie liczbowe dla danych, które są zgodne z ostatnim polem *RF2NVD* ; nie ma ono zastosowania do danych liczbowych w samej strukturze *MQRFH2* .

W wywołaniu *MQPUT* lub *MQPUT1* aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest *ENNAT*.

RF2FLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Należy podać następującą wartość:

RFNONE

Brak flag.

Wartością początkową tego pola jest *RFNONE*.

RF2FMT (8-bajtowy łańcuch znaków)

Formatuj nazwę danych, które są następujące po ostatnim polu *RF2NVD* .

Określa nazwę formatu danych, które są zgodne z ostatnim polem *RF2NVD* .

W wywołaniu *MQPUT* lub *MQPUT1* aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze *MQMD*.

Wartością początkową tego pola jest *FMNONE*.

RF2LEN (10-cyfrowa liczba całkowita ze znakiem)

Łączna długość *MQRFH2* łącznie ze wszystkimi polami *RF2NVL* i *RF2NVD* .

Jest to długość w bajtach struktury *MQRFH2* , w tym pola *RF2NVL* i *RF2NVD* na końcu struktury. Ważne jest, aby na końcu struktury było wiele par pól *RF2NVL* i *RF2NVD* , w kolejności:

```
length1, data1, length2, data2, ...
```

RF2LEN nie zawiera żadnych danych użytkownika, które mogą być zgodne z ostatnim polem *RF2NVD* na końcu struktury.

Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, należy rozważyć użycie *RF2LEN* jako wielokrotnej liczby czterech.

Następująca stała daje długość stałej części struktury, to znaczy długości wykluczając pola *RF2NVL* i *RF2NVD* :

RFLN2

Długość stałej części struktury *MQRFH2* .

Początkowa wartość tego pola to *RFLN2*.

RF2NVC (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków *RF2NVD*.

Ten parametr określa identyfikator kodowanego zestawu znaków dla danych w polu *RF2NVD* . Różni się on od zestawu znaków innych łańcuchów w strukturze *MQRFH2* i może różnić się od zestawu znaków danych (jeśli istnieją), które są następujące po ostatnim polu *RF2NVD* na końcu struktury.

RF2NVC musi mieć jedną z następujących wartości:

CCSID	Znaczenie
1200	Otwarto UCS-2
13488	Podzbiór UCS-2 2.0
17584	Podzbiór UCS-2 2.1 (zawiera symbol Euro)

CCSID	Znaczenie
1208	UTF-8

W przypadku zestawów znaków UCS-2 kodowanie (kolejność bajtów) *RF2NVD* musi być takie samo, jak kodowanie innych pól w strukturze *MQRFH2*. Znaki zastępcze (X'D800'przez X'DFFF') nie są obsługiwane.

Uwaga: Jeśli produkt *RF2NVC* nie ma jednej z wymienionych wcześniej wartości, a struktura *MQRFH2* wymaga konwersji w wywołaniu *MQGET*, wywołanie kończy się kodem przyczyny RC2111, a komunikat jest zwracany bez konwersji.

Wartość początkowa tego pola to 1208.

RF2NVD (łańcuch znaków n-byte)

Nazwa/wartość danych.

Jest to łańcuch znaków o zmiennej długości zawierający dane zakodowane przy użyciu składni typu XML. Długość w bajtach tego łańcucha jest udostępniana przez pole *RF2NVL*, które poprzedza pole *RF2NVD*. Długość ta powinna być wielokrotnością liczby czterech.

Pola *RF2NVL* i *RF2NVD* są opcjonalne, ale jeśli występują, muszą one występować jako para i być sąsiadującymi. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ pola te są opcjonalne, są one pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

RF2NVD jest nietypowy, ponieważ *nie* jest przekształcany w zestaw znaków określony w wywołaniu *MQGET*, gdy komunikat jest pobierany z opcją *GMCONV* w efekcie; *RF2NVD* pozostaje w oryginalnym zestawie znaków. Jednak program *RF2NVD* jest konwertowany na kodowanie określone w wywołaniu *MQGET*.

Składnia danych nazwa/wartość: Łańcuch składa się z jednego folderu zawierającego zero lub więcej właściwości. Folder jest ograniczany przez znaczniki początkowe i końcowe XML o takiej samej nazwie, jak folder:

```
<folder> property1 property2 ... </folder>
```

Znaki następujące po znaczniku końcowym folderu, do długości zdefiniowanej przez *RF2NVL*, muszą być puste. W obrębie folderu każda właściwość składa się z nazwy i wartości oraz opcjonalnie typu danych:

```
<name dt="datatype">value</name>
```

W poniższych przykładach:

- Znaki ogranicznika (<, =, "/", i >) muszą być określone dokładnie tak, jak pokazano.
- *name* jest nazwą właściwości określoną przez użytkownika. Więcej informacji na temat nazw zawiera poniższy przykład.
- *datatype* jest opcjonalnym typem danych określonym przez użytkownika. Patrz poniższy przykład dla poprawnych typów danych.
- *value* jest wartością właściwości określoną przez użytkownika. Więcej informacji na temat wartości można znaleźć w poniższych akapitach.
- Odstępy są znaczące między znakiem >, który poprzedza wartość, a znakiem <, który podąża za wartością, a co najmniej jedno puste musi poprzedzać dt=. Puste miejsca mogą być dowolnie kodowane między znacznikami, lub poprzedzającymi lub następującymi znacznikami (na przykład w celu poprawienia czytelności); te odstępy nie są znaczące.

Jeśli właściwości są ze sobą powiązane, można je grupować, ujmując je w znaczniki początkowe i końcowe XML o tej samej nazwie, co grupa:

```
<folder> <group> property1 property2 ... </group> </folder>
```

Grupy mogą być zagnieżdżone w innych grupach, bez ograniczeń, a grupa może występować więcej niż jeden raz w folderze. Jest on również poprawny dla folderu, który może zawierać niektóre właściwości w grupach i inne właściwości nie w grupach.

Nazwy właściwości, grup i folderów: Nazwy właściwości, grup i folderów muszą być poprawnymi nazwami znaczników XML, z wyjątkiem znaków dwukropka, które nie są dozwolone w nazwie właściwości, grupy lub folderu. W szczególności:

- Nazwy muszą zaczynać się od litery lub znaku podkreślenia. Poprawne litery są zdefiniowane w specyfikacji XML W3C i składają się głównie z kategorii: Ll, Lu, Lo, Lt i Nl.
- Pozostałe znaki w nazwie mogą być literami, cyframi dziesiętnymi, znakami podkreślenia, łącznikiem lub kropkami. Odpowiadają one kategoriom Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm, and Nd.
- Znaki zgodności Unicode (X'F900' i nowsze) nie są dozwolone w żadnej części nazwy.
- Nazwy nie mogą rozpoczynać się od łańcucha XML w żadnej mieszance wielkich lub małych liter.

Ponadto:

- W nazwach jest rozróżniana wielkość liter. Na przykład: ABC, abci Abc to trzy różne nazwy.
- Każdy folder ma oddzielną przestrzeń nazw. W wyniku tego grupa lub właściwość w jednym folderze nie powoduje konfliktu z grupą lub właściwością o tej samej nazwie w innym folderze.
- Grupy i właściwości zajmują tę samą przestrzeń nazw w folderze. W wyniku tego właściwość nie może mieć takiej samej nazwy, jak nazwa grupy w folderze, w którym znajduje się ta właściwość.

Ogólnie programy, które analizują pole *RF2NVD*, powinny ignorować właściwości lub grupy, które mają nazwy, których program nie rozpoznaje, pod warunkiem, że te właściwości lub grupy są poprawnie utworzone.

Typy danych właściwości: Każda właściwość może mieć opcjonalny typ danych. Jeśli zostanie podany, typ danych musi być jedną z następujących wartości, w przypadku wielkich, małych i małych liter:

Typ danych	Zastosowanie
string	Dowolna sekwencja znaków. Niektóre znaki muszą być określone za pomocą sekwencji o zmienionym znaczeniu.
boolean	Znak 0 lub 1 (1 oznacza TRUE).
bin.hex	Cyfrы szesnastkowe reprezentujące oktety.
i1	Liczba całkowita z zakresu od -128 do +127, wyrażona przy użyciu tylko cyfr dziesiętnych i opcjonalnego znaku.
i2	Liczba całkowita z zakresu od -32 768 do +32 767, wyrażona przy użyciu tylko cyfr dziesiętnych i opcjonalnego znaku.
i4	Liczba całkowita z zakresu od -2 147 483 648 do + 2 147 483 647, wyrażona przy użyciu tylko cyfr dziesiętnych i opcjonalnego znaku.
i8	Liczba całkowita z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, wyrażona przy użyciu tylko cyfr dziesiętnych i opcjonalnego znaku.
int	Liczba całkowita z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, wyrażona przy użyciu tylko cyfr dziesiętnych i opcjonalnego znaku. Może to być używane w lokalizacji i1, i2, i4 lub i8, jeśli nadawca nie chce sugerować konkretną precyzję.

Typ danych	Zastosowanie
<code>r4</code>	Liczba zmiennopozycyjna o wielkości z zakresu od 1.175E-37 do 3.402 823 47E+38, wyrażona za pomocą cyfr dziesiętnych, opcjonalnego znaku, opcjonalnej części ułamkowej i opcjonalnego wykładnika.
<code>r8</code>	Liczba zmiennopozycyjna o wielkości z zakresu od 2.225E-307 do 1.797 693 134 862 3E+308 wyrażona przy użyciu cyfr dziesiętnych, opcjonalnego znaku, opcjonalnej części ułamkowej i opcjonalnego wykładnika.

Wartości właściwości: wartość właściwości może składać się z dowolnych znaków, z wyjątkiem tych, które znajdują się na poniższym rysunku. Każde wystąpienie w wartości znaku oznaczonej jako "obowiązkowe" musi zostać zastąpione odpowiednią sekwencją o zmienionym znaczeniu. Każde wystąpienie w wartości znaku oznaczonej jako "opcjonalne" może zostać zastąpione przez odpowiednią sekwencję o zmienionym znaczeniu, ale nie jest to wymagane.

Znak	Sekwencja zmiany znaczenia	Użycie
&	<code>&amp;</code>	Obowiązkowe
<	<code><</code>	Obowiązkowe
>	<code>&gt;</code>	Opcjonalne
"	<code>&quot;</code>	Opcjonalne
'	<code>&apos;</code>	Opcjonalne

Uwaga: Znak & na początku sekwencji o zmienionym znaczeniu musi *nie* być zastępowany przez `&`.

W poniższym przykładzie odstęp w wartości są znaczące, jednak nie są potrzebne żadne sekwencje o zmienionym znaczeniu:

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (10-cyfrowa liczba całkowita ze znakiem)

Długość `RF2NVD`.

Określa długość danych w bajtach w polu `RF2NVD`. Aby uniknąć problemów z konwersją danych (jeśli istnieją), które *następuje* w polu `RF2NVD`, wartość `RF2NVL` powinna być wielokrotnością liczby czterech.

Uwaga: Pola `RF2NVL` i `RF2NVD` są opcjonalne, ale jeśli występują, muszą one występować jako para i być sąsiadującymi. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ pola te są opcjonalne, są one pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

RF2SID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

RFSIDV

Identyfikator reguł i struktury nagłówka formatowania.

Wartością początkową tego pola jest RFSIDV.

RF2VER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RFVER2

Reguły Version-2 i struktura nagłówka formatowania.

Początkowa wartość tego pola to RFVER2.

Wartości początkowe

Tabela 159. Początkowe wartości pól w tabeli MQRFH2		
Nazwa pola	Nazwa stałej	Wartość stałej
RF2SID	RFSIDV	'RFH¬'
RF2VER	RFVER2	2
RF2LEN	RFLen2	36
RF2ENC	ENNAT	Zależy od środowiska
RF2CSI	CINHT	-2
RF2FMT	FMNONE	Puste
RF2FLG	RFNONE	0
RF2NVC	Brak	1208
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRFH2 Structure
D*
D* Structure identifier
D RF2SID          1      4      INZ('RFH ')
D* Structure version number
D RF2VER          5      8I 0 INZ(2)
D* Total length of MQRFH2 including allNameValueLength and
D* NameValueDatafields
D RF2LEN          9      12I 0 INZ(36)
D* Numeric encoding of data that followslast NameValueData field
D RF2ENC          13     16I 0 INZ(273)
D* Character set identifier of data thatfollows last NameValueData field
D RF2CSI          17     20I 0 INZ(-2)
D* Format name of data that follows lastNameValueData field
D RF2FMT          21     28      INZ('      ')
D* Flags
D RF2FLG          29     32I 0 INZ(0)
D* Character set identifier ofNameValueData
D RF2NVC          33     36I 0 INZ(1208)
```

MQRMH-nagłówek komunikatu odwołania

Struktura MQRMH definiuje format nagłówka komunikatu odwołania.

Przegląd

Przeznaczenie: Ten nagłówek jest używany z wyjściami kanału komunikatów napisanych przez użytkownika w celu wysyłania dużych ilości danych (nazywanych "danymi masowymi"). z jednego menedżera kolejek do innego. Różnica w porównaniu z normalnym przesyłaniem komunikatów polega na tym, że dane masowe nie są zapisywane w kolejce. Zamiast tego w kolejce przechowywane są tylko

odwołanie do danych masowych. Zmniejsza to możliwość wyczerpania zasobów IBM MQ przez kilka dużych komunikatów.

Nazwa formatu: FMRMH.

Zestaw znaków i kodowanie: Dane znakowe w MQRMH oraz łańcuchy adresowane przez pola przesunięcia muszą znajdować się w zestawie znaków lokalnego menedżera kolejek. Ten atrybut jest nadawany przez atrybut menedżera kolejek produktu *CodedCharSetId*. Dane liczbowe w MQRMH muszą znajdować się w rodzimym kodowaniu komputera; jest to nadawane przez wartość ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQRMH muszą być ustawione w polach *MDCSI* i *MDENC* w:

- MQMD (jeśli struktura MQRMH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQRMH (wszystkie inne obserwacje).

Użycie: aplikacja umieszcza komunikat składający się z wywołania MQRMH, ale pomija dane masowe. Gdy komunikat jest odczytywany z kolejki transmisji przez agenta kanału komunikatów (MCA), wywoływany przez użytkownika program obsługi wyjścia komunikatów jest wywoływany w celu przetworzenia nagłówka komunikatu odwołania. Wyjście może dopisać do komunikatu referencyjnego dane masowe identyfikowane przez strukturę MQRMH, zanim agent MCA wyśle komunikat przez kanał do następnego menedżera kolejek.

Po zakończeniu odbierania komunikatu wyjścia komunikatu, który oczekuje na komunikaty odniesienia, powinien istnieć. Po odebraniu komunikatu referencyjnego wyjście powinno utworzyć obiekt na podstawie danych masowych, które są następujące po komunikacie MQRMH w komunikacie, a następnie przekazać komunikat odwołania bez danych masowych. Komunikat referencyjny może zostać później pobrany przez aplikację odczydując komunikat odniesienia (bez danych masowych) z kolejki.

Zwykle struktura MQRMH jest wszystkim, co znajduje się w komunikacie. Jeśli jednak komunikat znajduje się w kolejce transmisji, jeden lub więcej dodatkowych nagłówków poprzedzi strukturę MQRMH.

Komunikat odniesienia może również zostać wysłany do listy dystrybucyjnej. W tym przypadku struktura MQDH i powiązane z nią rekordy poprzedzają strukturę MQRMH, gdy komunikat znajduje się w kolejce transmisji.

Uwaga: Komunikat odniesienia nie powinien być wysyłany jako segmentowany komunikat, ponieważ wyjście komunikatu nie może przetworzyć tego komunikatu poprawnie.

- [“Konwersja danych” na stronie 1219](#)
- [“Pola” na stronie 1219](#)
- [“Wartości początkowe” na stronie 1224](#)
- [“Deklaracja RPG” na stronie 1225](#)

Konwersja danych

W celu konwersji danych konwersja struktury MQRMH obejmuje konwersję danych środowiska źródłowego, nazwy obiektu źródłowego, danych środowiska docelowego i nazwy obiektu docelowego. Wszystkie pozostałe bajty w bajtach *RMLen* początku struktury są odrzucane lub mają niezdefiniowane wartości po konwersji danych. Dane masowe zostaną przekształcone pod warunkiem, że spełnione są wszystkie poniższe warunki:

- Dane masowe są obecne w komunikacie, gdy wykonywana jest konwersja danych.
- Pole *RMFMT* w tabeli MQRMH ma wartość inną niż FMNONE.
- W przypadku wyjścia konwersji danych napisanych przez użytkownika istnieje określona nazwa formatu.

Należy jednak pamiętać, że zazwyczaj dane masowe *nie* są obecne w komunikacie, gdy komunikat znajduje się w kolejce, a w rezultacie dane masowe nie będą przekształcane przez opcję GMCONV.

Pola

Struktura MQRMH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

RMCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych masowych.

Określa identyfikator zestawu znaków danych masowych; nie ma zastosowania do danych znakowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych po tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest *ATBRKR*.

Wartością początkową tego pola jest *CSUNDF*.

RMDEL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych środowiska docelowego.

Jeśli to pole ma wartość zero, dane środowiska docelowego nie są dostępne, a parametr *RMDEO* jest ignorowany.

RMDEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych środowiska docelowego.

To pole służy do określania przesunięcia danych środowiska docelowego z początku struktury MQRMH. Dane środowiska docelowego mogą być określone przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Na przykład, dane środowiska docelowego mogą być ścieżką do katalogu obiektu, w którym mają być przechowywane dane masowe. Jeśli jednak twórca nie zna danych środowiska docelowego, jest on odpowiedzialny za wyjście komunikatów dostarczone przez użytkownika w celu określenia wszelkich potrzebnych informacji o środowisku.

Długość danych środowiska docelowego jest podawana przez produkt *RMDEL* ; Jeśli ta długość wynosi zero, nie ma danych środowiska docelowego, a program *RMDEO* jest ignorowany. Jeśli istnieje, dane środowiska docelowego muszą znajdować się całkowicie w bajtach *RMLen* od początku struktury.

Aplikacje nie powinny zakładać, że dane środowiska docelowego są ciągłe w przypadku danych adresowanych przez pola *RMSEO*, *RMSNOi* *RMDNO* .

Wartością początkową tego pola jest 0.

RMDL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych masowych.

Pole *RMDL* określa długość danych masowych, do których odwołuje się struktura MQRMH.

Jeśli w komunikacie znajdują się dane masowe, dane zaczynają się od przesunięcia w bajtach *RMLen* od początku struktury MQRMH. Długość całego komunikatu pomniejszona o *RMLen* określa długość danych masowych.

Jeśli dane są obecne w komunikacie, *RMDL* określa ilość danych, które są istotne. Normalny przypadek dotyczy wartości *RMDL* , która ma taką samą wartość jak długość danych znajdujących się w komunikacie.

Jeśli struktura MQRMH reprezentuje pozostałe dane w obiekcie (począwszy od określonego przesunięcia logicznego), wartość zero może być użyta dla *RMDL* , jeśli dane masowe nie są obecne w komunikacie.

Jeśli nie ma żadnych danych, koniec komunikatu MQRMH jest zbieżny z końcem komunikatu.

Wartością początkową tego pola jest 0.

RMDNL (10-cyfrowa liczba całkowita ze znakiem)

Długość nazwy obiektu docelowego.

Jeśli to pole ma wartość zero, nie ma nazwy obiektu docelowego, a parametr *RMDNO* jest ignorowany.

RMDNO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie nazwy obiektu docelowego.

To pole określa przesunięcie nazwy obiektu docelowego od początku struktury MQRMH. Nazwa obiektu docelowego może być określona przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu docelowego, jest on odpowiedzialny za wyjście z komunikatu dostarczonego przez użytkownika w celu zidentyfikowania obiektu, który ma zostać utworzony lub zmodyfikowany.

Długość nazwy obiektu docelowego jest podawana przez produkt *RMDNL* ; Jeśli ta długość wynosi zero, nie istnieje nazwa obiektu docelowego, a parametr *RMDNO* jest ignorowany. Jeśli ta wartość jest obecna, nazwa obiektu docelowego musi znajdować się całkowicie w bajtach *RMLN* od początku struktury.

Aplikacje nie powinny zakładać, że nazwa obiektu docelowego jest ciągła z dowolnym z danych adresowanych przez pola *RMSEO*, *RMSNOi* *RMDEO* .

Wartością początkową tego pola jest 0.

RMDO (10-cyfrowa liczba całkowita ze znakiem)

Niskie przesunięcie danych masowych.

To pole określa niską wartość przesunięcia danych masowych od początku obiektu, którego część stanowi część danych masowych. Przesunięcie danych masowych od początku obiektu jest nazywane *przesuniętym logicznym*. Jest to *nie* przesunięcie fizyczne danych masowych od początku struktury MQRMH-przesunięcie to jest nadawane przez produkt *RMLN*.

Aby umożliwić wysyłanie dużych obiektów za pomocą komunikatów referencyjnych, przesunięcie logiczne jest podzielone na dwa pola, a rzeczywiste przesunięcie logiczne jest nadawane przez sumę tych dwóch pól:

- *RMDO* reprezentuje pozostałą część otrzymaną, gdy przesunięcie logiczne dzieli się na 1 000 000 000. Jest to zatem wartość z zakresu od 0 do 999 999 999.
- *RMDO2* reprezentuje wynik uzyskany, gdy przesunięcie logiczne dzieli się na 1 000 000 000. Jest to zatem liczba pełnych wielokrotności 1 000 000 000 istniejących w logice offsetowej. Liczba wielokrotności mieści się w zakresie od 0 do 999 999 999.

Wartością początkową tego pola jest 0.

RMDO2 (10-cyfrowa liczba całkowita ze znakiem)

Wysokie przesunięcie danych masowych.

To pole określa wysokie przesunięcie danych masowych od początku obiektu, którego część stanowi część danych masowych. Jest to wartość z zakresu od 0 do 999 999 999. Szczegółowe informacje można znaleźć w sekcji *RMDO*.

Wartością początkową tego pola jest 0.

RMENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych masowych.

Określa kodowanie numeryczne danych masowych; nie ma zastosowania do danych liczbowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest ENNAT.

RMFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatu odwołania.

Zdefiniowane są następujące opcje:

RMLAST

Komunikat referencyjny zawiera lub reprezentuje ostatnią część obiektu.

Ta opcja wskazuje, że komunikat odniesienia reprezentuje ostatnią część obiektu, do którego istnieje odwołanie.

RMNLST

Komunikat odniesienia nie zawiera ani nie reprezentuje ostatniej części obiektu.

RMNLST jest zdefiniowane w dokumentacji programu pomocy. Ta opcja nie jest przeznaczona do użycia z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest RMNLST.

RMFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych masowych.

Określa nazwę formatu danych masowych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

RMLEN (10-cyfrowa liczba całkowita ze znakiem)

Całkowita długość MQRMH, w tym łańcuchy na końcu pól statycznych, ale nie dane masowe.

Początkowa wartość tego pola wynosi zero.

RMOII (24-bajtowy łańcuch bitowy)

Identyfikator instancji obiektu.

To pole może być używane do identyfikowania konkretnej instancji obiektu. Jeśli nie jest to potrzebne, należy ustawić następującą wartość:

OIINON

Nie określono identyfikatora instancji obiektu.

Wartość jest binarna zero dla długości pola.

Długość tego pola jest podana przez parametr LNOIID. Wartością początkową tego pola jest OIINON.

RMOT (8-bajtowy łańcuch znaków)

Typ obiektu.

Jest to nazwa, która może być używana przez wyjście komunikatu do rozpoznawania typów komunikatów referencyjnych obsługiwanych przez ten komunikat. Należy rozważyć, aby nazwa była zgodna z tymi samymi regułami, co w polu *RMFMT*.

Początkowa wartość tego pola wynosi 8 znaków odstępu.

RMSEL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych środowiska źródłowego.

Jeśli to pole ma wartość zero, nie ma danych środowiska źródłowego, a program *RMSEO* jest ignorowany.

Wartością początkową tego pola jest 0.

RMSEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych środowiska źródłowego.

To pole określa przesunięcie źródła danych środowiska źródłowego od początku struktury MQRMH. Dane środowiska źródłowego mogą być określone przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Na przykład źródłem danych środowiska źródłowego może być ścieżka do katalogu obiektu zawierającego dane masowe. Jeśli jednak twórca nie zna danych dotyczących środowiska źródłowego, odpowiedzialność za wyjście komunikatów dostarczonych przez użytkownika w celu określenia wszelkich wymaganych informacji o środowisku.

Długość danych środowiska źródłowego jest podawana przez produkt *RMSEL* ; Jeśli ta długość wynosi zero, nie ma danych środowiska źródłowego, a program *RMSEO* jest ignorowany. Jeśli jest to obecne, dane środowiska źródłowego muszą znajdować się całkowicie w bajtach *RMLen* od początku struktury.

Aplikacje nie powinny zakładać, że dane o środowisku zaczynają się od razu po ostatnim stałym polu w strukturze lub że jest ona ciągła z dowolnymi danymi adresowane przez pola *RMSNO*, *RMDEOi* *RMDNO* .

Wartością początkową tego pola jest 0.

RMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

RMSIDV

Identyfikator struktury nagłówka komunikatu odniesienia.

Wartością początkową tego pola jest RMSIDV.

RMSNL (10-cyfrowa liczba całkowita ze znakiem)

Długość nazwy obiektu źródłowego.

Jeśli to pole ma wartość zero, nie istnieje żadna nazwa obiektu źródłowego, a parametr *RMSNO* jest ignorowany.

Wartością początkową tego pola jest 0.

RMSNO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie nazwy obiektu źródłowego.

To pole określa przesunięcie nazwy obiektu źródłowego od początku struktury MQRMH. Nazwa obiektu źródłowego może zostać określona przez twórcę komunikatu referencyjnego, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu źródłowego, to jest on odpowiedzialny za wyjście z komunikatu dostarczonego przez użytkownika w celu zidentyfikowania obiektu, do którego ma być uzyskany dostęp.

Długość nazwy obiektu źródłowego jest podawana przez produkt *RMSNL* ; Jeśli ta długość wynosi zero, nie istnieje żadna nazwa obiektu źródłowego, a parametr *RMSNO* jest ignorowany. Jeśli ta opcja jest obecna, nazwa obiektu źródłowego musi znajdować się całkowicie w bajtach *RMLen* od początku struktury.

Aplikacje nie powinny zakładać, że nazwa obiektu źródłowego jest ciągła z dowolnym z danych adresowanych przez pola *RMSEO*, *RMDEOi* *RMDNO* .

Wartością początkową tego pola jest 0.

RMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RMVER1

Struktura nagłówka komunikatu odwołania Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

RMVERC

Bieżąca wersja struktury nagłówka komunikatu odwołania.

Początkowa wartość tego pola to RMVER1.

Wartości początkowe

Tabela 160. Początkowe wartości pól w MQRMH		
Nazwa pola	Nazwa stałej	Wartość stałej
RMSID	RMSIDV	'RMH¬'
RMVER	RMVER1	1
RMLen	Brak	0
RMENC	ENNAT	Zależy od środowiska
RMCSI	CSUNDF	0
RMFMT	FMNONE	Puste
RMFLG	RMNLST	0
RMOT	Brak	Puste
RMOII	OIINON	Wartości null
RMSEL	Brak	0
RMSEO	Brak	0
RMSNL	Brak	0
RMSNO	Brak	0
RMDEL	Brak	0
RMDEO	Brak	0
RMDNL	Brak	0
RMDNO	Brak	0
RMDL	Brak	0
RMDO	Brak	0
RMDO2	Brak	0
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

```

D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRMH Structure
D*
D* Structure identifier
D RMSID          1      4    INZ('RMH ')
D* Structure version number
D RMVER          5      8I 0 INZ(1)
D* Total length of MQRMH, including strings at end of fixed fields, but not
D* the bulk data
D RMLen          9      12I 0 INZ(0)
D* Numeric encoding of bulk data
D RMENC          13     16I 0 INZ(273)
D* Character set identifier of bulk data
D RMCSI          17     20I 0 INZ(0)
D* Format name of bulk data
D RMFMT          21     28    INZ('      ')
D* Reference message flags
D RMFLG          29     32I 0 INZ(0)

```

```

D* Object type
D RMOT 33 40 INZ
D* Object instance identifier
D RMOII 41 64 INZ(X'0000000000000000-
D 000000000000000000000000-
D 000000000000')
D* Length of source environmentdata
D RMSEL 65 68I 0 INZ(0)
D* Offset of source environmentdata
D RMSEO 69 72I 0 INZ(0)
D* Length of source object name
D RMSNL 73 76I 0 INZ(0)
D* Offset of source object name
D RMSNO 77 80I 0 INZ(0)
D* Length of destination environmentdata
D RMDEL 81 84I 0 INZ(0)
D* Offset of destination environmentdata
D RMDEO 85 88I 0 INZ(0)
D* Length of destination objectname
D RMDNL 89 92I 0 INZ(0)
D* Offset of destination objectname
D RMDNO 93 96I 0 INZ(0)
D* Length of bulk data
D RMDL 97 100I 0 INZ(0)
D* Low offset of bulk data
D RMDO 101 104I 0 INZ(0)
D* High offset of bulk data
D RMDO2 105 108I 0 INZ(0)

```

Deklaracja RPG

MQRR-rekord odpowiedzi

Struktura MQRR jest używana do odbierania kodu zakończenia i kodu przyczyny wynikającego z operacji otwierania lub umieszczania dla jednej kolejki docelowej, gdy miejscem docelowym jest lista dystrybucyjna.

Przegląd

Cel: MQRR jest strukturą wyjściową dla wywołań MQOPEN, MQPUT i MQPUT1 .

Zestaw znaków i kodowanie: Dane w tabeli MQRR muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ , struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

Użycie: udostępniając tablicę tych struktur w wywołaniach MQOPEN i MQPUT lub w wywołaniu MQPUT1 , możliwe jest określenie kodów zakończenia i kodów przyczyny dla wszystkich kolejek na liście dystrybucyjnej, gdy wynik wywołania jest mieszany, to znaczy, gdy wywołanie powiedzie się dla niektórych kolejek na liście, ale dla innych nie powiedzie się. Kod przyczyny RC2136 z wywołania wskazuje, że rekordy odpowiedzi (jeśli zostały udostępnione przez aplikację) zostały ustawione przez menedżer kolejek.

- [“Pola” na stronie 1225](#)
- [“Wartości początkowe” na stronie 1226](#)
- [“Deklaracja RPG” na stronie 1226](#)

Pola

Struktura MQRR zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

RRCC (10-cyfrowa liczba całkowita ze znakiem)

Kod zakończenia dla kolejki.

Jest to kod zakończenia wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest CCOK.

RRREA (10-cyfrowa liczba całkowita ze znakiem)

Kod przyczyny dla kolejki.

Jest to kod przyczyny wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionych w wywołaniu MQOPEN lub MQPUT1.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest RCNONE.

Wartości początkowe

Tabela 161. Początkowe wartości pól w MQRR		
Nazwa pola	Nazwa stałej	Wartość stałej
RRCC	CCOK	0
RRREA	RCBRAK	0

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D*
D* MQRR Structure
D*
D* Completion code for queue
D  RRCC              1          4I 0 INZ(0)
D* Reason code for queue
D  RRRRA             5          8I 0 INZ(0)
```

MQSCO-opcje konfiguracji protokołu SSL

Struktura MQSCO (z polami SSL w strukturze MQCD) umożliwia uruchomienie aplikacji jako IBM MQ MQI client w celu określenia opcji konfiguracyjnych, które sterują użyciem protokołu SSL dla połączenia klienckiego, gdy protokół kanału jest protokołem TCP/IP.

Przegląd

Cel: Struktura jest parametrem wejściowym w wywołaniu MQCONN.

Jeśli protokół kanału dla kanału klienta nie jest protokołem TCP/IP, struktura MQSCO jest ignorowana.

Zestaw znaków i kodowanie: Dane w tabeli MQSCO muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT.

- [“Pola” na stronie 1226](#)
- [“Wartości początkowe” na stronie 1230](#)
- [“Deklaracja RPG” na stronie 1231](#)

Pola

Struktura MQSCO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

SCAIC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba rekordów informacji uwierzytelniających (MQAIR), do których adresowane są pola SCAIP lub SCAIO . Więcej informacji na ten temat zawiera sekcja [“MQAIR-rekord informacji uwierzytelniającej”](#) na stronie 1027. Wartość musi być równa zero lub większa. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2383.

To jest pole wejściowe. Wartością początkową tego pola jest 0.

SCAIO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach pierwszego rekordu informacji uwierzytelniających od początku struktury MQSCO. Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli wartość SCAIC wynosi zero.

Aby określić rekordy MQAIR, ale nie oba, można użyć opcji SCAIO lub SCAIP , aby uzyskać szczegółowe informacje, należy zapoznać się z opisem pola SCAIP .

To jest pole wejściowe. Wartością początkową tego pola jest 0.

SCAIP (10-cyfrowa liczba całkowita ze znakiem)

Jest to adres pierwszego rekordu informacji uwierzytelniających. Pole jest ignorowane, jeśli wartość SCAIC wynosi zero.

Tablicę rekordów MQAIR można udostępnić na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika SCAIP

W takim przypadku aplikacja może zadeklarować tablicę rekordów MQAIR, która jest oddzielona od struktury MQSCO, i ustawić parametr SCAIP na adres tablicy.

Należy rozważyć użycie produktu SCAIP dla języków programowania, które obsługują typ danych wskaźnika w sposób przenośny dla różnych środowisk (na przykład język programowania w języku C).

- Za pomocą pola przesunięcia SCAIO

W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą obiekt MQSCO, po którym następuje tablica rekordów MQAIR, a następnie ustawić parametr SCAIO na przesunięcie pierwszego rekordu w tablicy od początku struktury MQSCO. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być zakwaterowana w tabeli MQLONG (najbardziej restrykcyjnym językiem programowania jest język COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

Należy rozważyć użycie produktu SCAIO w językach programowania, które nie obsługują typu danych wskaźnika, lub które implementują typ danych wskaźnika w sposób, który nie jest przenośny dla różnych środowisk (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki, można użyć tylko jednej z następujących opcji: SCAIP i SCAIO . Wywołanie nie powiedzie się, kod przyczyny RC2384 , jeśli oba są niezerowe.

To jest pole wejściowe. Wartością początkową tego pola jest pusty wskaźnik w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie łańcuch bajtowy all-null.

Uwaga: W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

SCCERLBL (10-cyfrowa liczba całkowita ze znakiem)

W tym polu podano szczegółowe informacje na temat używanej etykiety certyfikatu.

Program IBM MQ inicjuje wartość pola SCCERLBL jako odstępy. Wprowadź wymaganą wartość lub zaakceptuj wartość domyślną.

ibmwebspheremq<user_id> jest poprawną wartością dla tego pola dla wszystkich wersji produktu, a dla wersji MQSCO mniejszej niż 5.0 jest to jedyna poprawna wartość. W związku z tym wartość tego pola jest interpretowana w czasie wykonywania i w razie potrzeby zmieniana. Jeśli zostanie określona

wersja MQSCO mniejsza niż 5.0 lub zostanie zaakcepta domyślna wartość odstępu dla pola SCCERLBL, system użyje wartości `ibmwebspheremq<user_id>`.

To jest pole wejściowe.

SCCERTVPOL (10-cyfrowa liczba całkowita ze znakiem)

To pole określa typ strategii sprawdzania poprawności certyfikatu. Pole można ustawić na jedną z następujących wartości:

MQ_CERT_VAL_POLICY_ANY

Zastosuj każdą ze strategii sprawdzania poprawności certyfikatów obsługiwanych przez bibliotekę bezpiecznych gniazd. Zaakceptuj łańcuch certyfikatów, jeśli dowolna z strategii uzna, że łańcuch certyfikatów jest poprawny.

MQ_CERT_VAL_POLICY_RFC5280

Zastosuj tylko strategię sprawdzania poprawności certyfikatu zgodną ze standardem RFC5280 . To ustawienie zapewnia bardziej restrykcyjne sprawdzanie poprawności niż ustawienie ANY, ale odrzuca niektóre starsze certyfikaty cyfrowe.

Początkowa wartość tego pola to `MQ_CERT_VAL_POLICY_ANY`

SCCH (10-cyfrowa liczba całkowita ze znakiem)

To pole zawiera szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienckim.

Ustaw pole na łańcuch w następującym formacie lub pozostaw to pole puste lub puste:

```
GSK_PKCS11=<the PKCS #11 driver path and file name>;<the PKCS #11 token label>;<the PKCS #11 token password>;<symmetric cipher setting>;
```

Aby używać sprzętu szyfrującego zgodnego z interfejsem PKCS11 , na przykład IBM 4960 lub IBM 4963, należy określić ścieżkę do sterownika PKCS11 , etykietę znacznika PKCS11 oraz łańcuchy haseł znaczników PKCS11 , każde zakończone znakiem średnika.

Ścieżka do sterownika PKCS #11 jest pełną ścieżką do biblioteki współużytkowanej udostępniających obsługę karty PKCS #11 . Nazwa pliku sterownika PKCS #11 jest nazwą biblioteki współużytkowanej. Przykładem wartości wymaganej dla ścieżki i #11 pliku #11 PKCS jest:

```
/usr/lib/pkcs11/PKCS11_API.so
```

Etykieta znacznika PKCS #11 musi być całkowicie zapisana małymi literami. Jeśli sprzęt został skonfigurowany z małą lub wielką etykietą znacznika, należy zrekonfigurować je przy użyciu tej małej etykiety.

Jeśli nie jest wymagana żadna konfiguracja sprzętu szyfrującego, należy ustawić pole puste lub mieć wartość NULL.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Jeśli wartość ta nie jest poprawna lub wystąpi błąd podczas konfigurowania sprzętu szyfrującego, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2382.

To jest pole wejściowe. Długość tego pola jest podana przez LNSSCH. Wartością początkową tego pola jest puste znaki.

SCEPSUITEB (10-cyfrowa liczba całkowita ze znakiem)

To pole określa, czy używana jest kryptografia zgodna ze standardem Suite B, oraz jaki poziom siły jest używany. Wartość może być jedną lub większą z następujących wartości:

- SCEPSUITEB0

Kryptografia zgodna z pakietem B nie jest używana.

- SCEPSUITEB1

Używane są 128-bitowe zabezpieczenie mocy 128-bitowe Suite.

- SCEPSUITEB2

Pakiet B 192-bit bezpieczeństwa mocy jest używany.

Uwaga: Użycie parametru SCEPSUITEB0 z dowolną inną wartością w tym polu jest niepoprawne.

SCFR (10-cyfrowa liczba całkowita ze znakiem)

Produkt IBM MQ może być skonfigurowany z użyciem sprzętu szyfrującego, dzięki czemu używane moduły szyfrowania są dostarczane przez produkt sprzętowy. Te moduły mogą być certyfikowane zgodnie ze standardem FIPS do konkretnego poziomu w zależności od używanego produktu sprzętowego.

Użyj tego pola, aby określić, że tylko algorytmy certyfikowane przez FIPS są używane, jeśli kryptografia jest dostępna w oprogramowaniu IBM MQ.

Po zainstalowaniu produktu IBM MQ instalowana jest również implementacja szyfrowania SSL, która udostępnia moduły z certyfikatem FIPS.

Możliwe wartości to:

MQSSL_FIPS_NO

Jest to wartość domyślna. Po ustawieniu tej wartości:

- Można użyć dowolnego obiektu CipherSpec obsługiwane na konkretnej platformie.
- W przypadku uruchamiania bez użycia sprzętu szyfrującego, następujące specyfikacje CipherSpecs są uruchamiane za pomocą szyfrowania z certyfikatem FIPS 140-2 na platformach IBM MQ :
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES

W przypadku ustawienia tej wartości, o ile nie jest używany sprzęt szyfrujący do wykonania kryptografii, można mieć pewność, że

- W specyfikacji CipherSpec stosowane do tego połączenia klienta mogą być używane tylko algorytmy szyfrowania certyfikowane przez FIPS.
- Połączenia przychodzące i wychodzące kanału SSL są pomyślne tylko wtedy, gdy używany jest jeden z następujących specyfikacji szyfru:
 - TLS_RSA_WITH_3DES_EDE_CBC_SHA
 - TLS_RSA_WITH_AES_128_CBC_SHA
 - TLS_RSA_WITH_AES_256_CBC_SHA

Uwagi:

1. Klasa CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna.
2. Jeśli to możliwe, jeśli skonfigurowano tylko standard FIPS CipherSpecs , klient MQI odrzuca połączenia, które określają CipherSpec withRC2393 bez obsługi standardu FIPS. Produkt IBM MQ nie gwarantuje odrzucenia wszystkich takich połączeń i jest odpowiedzialny za określenie, czy konfiguracja produktu IBM MQ jest zgodna ze standardem FIPS.

SCKR (10-cyfrowa liczba całkowita ze znakiem)

To pole ma znaczenie tylko w przypadku produktu IBM MQ MQI clients działającego w systemach Systemy UNIX i Windows . Określa ono położenie pliku bazy danych kluczy, w którym są przechowywane klucze i certyfikaty. Plik bazy danych kluczy musi mieć nazwę pliku o nazwie zzz . kdb, gdzie zzz jest wybierany przez użytkownika. Pole SCKR zawiera ścieżkę do tego pliku wraz z trzonkiem nazwy pliku (wszystkie znaki w nazwie pliku, do których nie ma, ale nie zawiera finalnego . kdb). Przyrostek pliku . kdb jest dodawany automatycznie.

Każdy plik bazy danych kluczy ma powiązany *plik ukrytych haseł*. W ten sposób przechowywane są zaszyfrowane hasła, które umożliwiają programistyczny dostęp do bazy danych kluczy. Plik ukrytych haseł musi znajdować się w tym samym katalogu i mieć ten sam plik macierzysty, co baza danych kluczy, i musi kończyć się przyrostkiem .sth.

Na przykład, jeśli pole *SCKR* ma wartość /xxx/yyy/key, plikiem bazy danych kluczy musi być /xxx/yyy/key.kdb, a plikiem ukrytych haseł musi być /xxx/yyy/key.sth, gdzie xxx i yyy reprezentują nazwy katalogów.

Jeśli wartość jest krótsza niż długość pola, zakończ ją znakiem o kodzie zero lub dopełniaj ją spacjami do długości pola. Wartość nie jest sprawdzana. Jeśli wystąpił błąd podczas uzyskiwania dostępu do repozytorium kluczy, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2381.

Aby uruchomić połączenie SSL z IBM MQ MQI client, należy ustawić *SCKR* na poprawną nazwę pliku bazy danych kluczy.

To jest pole wejściowe. Długość tego pola jest podana przez LNSSKR. Wartość początkowa tego pola jest pustym znakiem.

SCSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

SCSIDV

Identyfikator struktury opcji konfiguracji protokołu SSL.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest SCSIDV.

SCVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

SCVER1

Struktura opcji konfiguracji protokołu SSL w wersji Version-1 .

SCVER2

Struktura opcji konfiguracji protokołu SSL w wersji Version-2 .

Następująca stała określa numer wersji bieżącej wersji:

SCVERC

Bieżąca wersja struktury opcji konfiguracji SSL.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to SCVER2

Wartości początkowe

Tabela 162. Początkowe wartości pól w MQSCO		
Nazwa pola	Nazwa stałej	Wartość stałej
SCSID	SCSIDV	'SC0~'
SCVER	SCVER5	1
SCKR	Brak	Pusty łańcuch lub odstępy
SCCH	Brak	Pusty łańcuch lub odstępy
SCAIC	Brak	0
SCAIO	Brak	0
SCAIP	Brak	Pusty wskaźnik lub zerowe bajty

Tabela 162. Początkowe wartości pól w MQSCO (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
SCKRC	Brak	Pusty wskaźnik lub zerowe bajty
SCFR	Brak	Pusty wskaźnik lub zerowe bajty
SCEPSUITEB	Brak	Pusty wskaźnik lub zerowe bajty
SCCERTVPOL	Brak	Pusty wskaźnik lub zerowe bajty
SCCERLBL	Brak	Pusty wskaźnik lub zerowe bajty
Uwagi: 1. Symbol – reprezentuje pojedynczy pusty znak. 2. Informacje na temat opcji SCEPSUITEB zawiera sekcja “Deklaracja RPG” na stronie 1231.		

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQSCO Structure
D*
D* Structure identifier
D  SCSID          1      4    INZ('SCO ')
D* Structure version number
D  SCVER          5      8I 0 INZ(1)
D* Location of SSL key repository
D  SCKR           9     264  INZ
D* Cryptographic hardware configuration string
D  SCCH          265     520  INZ
D* Number of MQAIR records present
D  SCAIC         521     524I 0 INZ(0)
D* Offset of first MQAIR record from start of MQSCO structure
D  SCAIO         525     528I 0 INZ(0)
D* Address of first MQAIR record
D  SCAIP         529     544*  INZ(*NULL)
D* Ver:1 **
D* Number of unencrypted bytes sent/received before secret key is
D* reset
D  SCKRC         545     548I 0 INZ(0)
D* Using FIPS-certified algorithms
D  SCFR         549     552I 0 INZ(0)
D* Ver:2 **
* Use only Suite B cryptographic algorithms
D  SCEPSUITEB0
D  SCEPSUITEB1     553     556I 0 INZ(1)
D  SCEPSUITEB2     557     560I 0 INZ(0)
D  SCEPSUITEB3     561     564I 0 INZ(0)
D  SCEPSUITEB4     565     568I 0 INZ(0)
D  SCEPSUITEB      10I 0 DIM(4) OVERLAY(SCEPSUITEB0)
D* Ver:3 **
D* Certificate validation policy
D  SCCERTVPOL     569     572I 0 INZ(0)
D* Ver:4 **
    
```

MQSD-deskryptor subskrypcji

Struktura MQSD służy do określania szczegółowych informacji na temat subskrybowanego subskrypcji.

Przegląd

Przeznaczenie

Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSUB.

Zarządzane subskrypcje

Jeśli aplikacja nie ma konkretnej potrzeby użycia określonej kolejki jako miejsca docelowego dla tych publikacji, które są zgodne z jej subskrypcją, może ona korzystać z funkcji subskrypcji zarządzanej. Jeśli aplikacja zdecyduje się na korzystanie z subskrypcji zarządzanej, menedżer kolejek informuje subskrybenta o miejscu docelowym, w którym są wysyłane opublikowane komunikaty, udostępniając uchwyt obiektu jako wyjście wywołania MQSUB. Więcej informacji na ten temat zawiera sekcja [HOBj \(10-digit signed integer\)-input/output](#).

Po usunięciu subskrypcji menedżer kolejek podejmuje również działania w celu wyczyszczenia komunikatów, które nie zostały pobrane z zarządzanego miejsca docelowego, w następujących sytuacjach:

- Po usunięciu subskrypcji-za pomocą komendy MQCLOSE z CORMSB-a zarządzany Hobj jest zamknięty.
- Domyślnie oznacza to, że połączenie zostało utracone do aplikacji przy użyciu nietrwałej subskrypcji (SONDUR)
- Po utracie ważności subskrypcji, ponieważ utraciła ważność, a zarządzany Hobj jest zamknięty.

Należy używać subskrypcji zarządzanych z subskrypcjami nietrwałymi, tak aby możliwe było czyszczenie oraz aby komunikaty dotyczące zamkniętych subskrypcji nietrwałych nie zajmował miejsca w menedżerze kolejek. Trwałe subskrypcje mogą również używać zarządzanych miejsc docelowych.

Zestaw znaków i kodowanie

Dane w tabeli MQSD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu *CodedCharSetId* oraz kodowanie lokalnego menedżera kolejek nadawanego przez ENNAT. Jeśli jednak aplikacja jest uruchomiona jako klient IBM MQ, struktura musi znajdować się w zestawie znaków i kodowaniu klienta.

- [“Pola” na stronie 1232](#)
- [“Wartości początkowe” na stronie 1245](#)
- [“Deklaracja RPG” na stronie 1246](#)

Pola

Struktura MQSD zawiera następujące pola: pola są opisane w kolejności alfabetycznej:

SDAID (32 bajtowy łańcuch znaków)

Ta wartość znajduje się w polu *MDAID* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *SDAID* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Więcej informacji na temat produktu *MDAID* zawiera sekcja [MDAID](#).

Jeśli opcja *SOSETI* nie jest określona, wartość *MDAID*, która jest ustawiona w każdym komunikacie opublikowanym dla tej subskrypcji, jest pusta, jako domyślne informacje o kontekście.

Jeśli zostanie podana opcja *SOSETI*, *SDAID* jest generowana przez użytkownika, a to pole jest polem wejściowym, które zawiera *MDAID*, które mają być ustawione w każdej publikacji dla tej subskrypcji.

Długość tego pola nadawana jest przez *LNAIDD*. Początkowa wartość tego pola to 32 znaki puste.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji *SOALT* *SDAID* mogą zostać zmienione wszystkie przyszłe komunikaty publikacji.

W przypadku powrotu z wywołania MQSUB przy użyciu *SORES* to pole jest ustawione na bieżącą wartość *MDAID* używaną dla subskrypcji.

SDACC (łańcuch znaków 32-bitowych)

Ta wartość znajduje się w polu *MDACC* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *MDACC* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja [Kontekst komunikatu](#).

Więcej informacji na temat produktu *MDACC* zawiera sekcja [MDACC](#).

Dla pola *SDACC* można użyć następującej wartości specjalnej:

ACNONE

Nie określono znacznika rozliczeniowego.

Wartość jest binarna zero dla długości pola.

Jeśli opcja *SOSETI* nie jest określona, to znacznik rozliczania jest generowany przez menedżer kolejek jako domyślne informacje kontekstowe, a to pole jest polem wyjściowym zawierającym *MDACC*, który jest ustawiany w każdym komunikacie opublikowanym dla tej subskrypcji.

Jeśli określono opcję *SOSETI*, użytkownik i to pole są generowane przez użytkownika, a pole to zawiera pole wejściowe, które zawiera *MDACC*, które mają być ustawione w każdej publikacji dla tej subskrypcji.

Długość tego pola jest podana przez *LNACCT*. Wartością początkową tego pola jest *ACNONE*.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji *SOALT* można zmienić wartość parametru *MDACC* w dowolnych przyszłych komunikatach publikowania.

W przypadku powrotu z wywołania *MQSUB* przy użyciu komendy *SORES*, to pole jest ustawione na bieżącą wartość *MDACC* używaną dla subskrypcji.

SDASI (40-bajtowy łańcuch bitowy)

Jest to identyfikator zabezpieczeń przekazywany razem z produktem *SDAU* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji.

Wartość *SDASI* jest używana tylko wtedy, gdy określono parametr *SOALTU*, a pole *SDAU* nie jest całkowicie puste do pierwszego znaku o kodzie zero lub do końca pola.

W przypadku powrotu z wywołania *MQSUB* przy użyciu komendy *SORE* to pole pozostaje niezmienione.

Więcej informacji na ten temat można znaleźć w opisie [ODASI](#) w typie danych *MQOD*.

SDAU (12-bajtowy łańcuch znaków)

Jeśli zostanie podana wartość *SOALTU*, to pole będzie zawierać alternatywny identyfikator użytkownika, który będzie używany do sprawdzania autoryzacji subskrypcji i danych wyjściowych do kolejki docelowej (określonej w parametrze *Hobj* wywołania *MQSUB*), w miejsce identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli operacja się powiedzie, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji w miejscu identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli określono parametr *SOALTU* i pole to jest całkowicie puste do pierwszego znaku o kodzie zero lub końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy nie jest wymagana autoryzacja użytkownika w celu zasubskrybowania tego tematu z opcjami określonymi lub kolejką docelową dla danych wyjściowych.

Jeśli parametr *SOALTU* nie jest określony, to pole jest ignorowane.

W przypadku powrotu z wywołania *MQSUB* przy użyciu komendy *SORE* to pole pozostaje niezmienione.

To jest pole wejściowe. Długość tego pola jest podana przez *LNUID*. Początkowa wartość tego pola to 12 znaków odstępu.

SDCID (24-bitowy łańcuch bitowy)

Wszystkie publikacje wystane w celu dopasowania do tej subskrypcji zawierają ten identyfikator korelacji w deskrypcji komunikatu. Jeśli wiele subskrypcji korzysta z tej samej kolejki w celu pobrania ich publikacji, użycie identyfikatora MQGET według identyfikatora korelacji umożliwia uzyskanie tylko tych publikacji, które mają zostać uzyskane. Ten identyfikator korelacji może być wygenerowany przez menedżera kolejek lub przez użytkownika.

Jeśli opcja S0SCID nie jest określona, to identyfikator korelacji jest generowany przez menedżer kolejek, a pole to jest polem wyjściowym zawierającym identyfikator korelacji ustawiony w każdym komunikacie opublikowanym dla tej subskrypcji.

Jeśli podano opcję S0SCID, identyfikator korelacji jest generowany przez użytkownika, a pole to jest polem wejściowym, które zawiera identyfikator korelacji, który ma być ustawiony w każdej publikacji dla tej subskrypcji. W tym przypadku, jeśli pole zawiera CINONE, identyfikatorem korelacji ustawionym w każdym komunikacie opublikowanym dla tej subskrypcji jest identyfikator korelacji, który został utworzony przez oryginał komunikatu umieszczonego w komunikacie.

Jeśli podano opcję S0GRP, a określony identyfikator korelacji jest taki sam, jak istniejąca grupowa subskrypcja przy użyciu tej samej kolejki i nakładającego się łańcucha tematu, tylko najbardziej znacząca subskrypcja w grupie jest udostępniana wraz z kopią publikacji.

Długość tego pola jest podawana przez LNCID. Wartością początkową tego pola jest CINONE.

Jeśli istniejąca subskrypcja zostanie zmieniona za pomocą opcji SOALT, a pole to jest polem wejściowym, to można zmienić identyfikator korelacji subskrypcji, chyba że subskrypcja została utworzona przy użyciu opcji S0GRP.

W przypadku powrotu z wywołania MQSUB przy użyciu komendy SORESto pole jest ustawiane na bieżący identyfikator korelacji dla subskrypcji.

SDEXP (10-cyfrowa liczba całkowita ze znakiem)

Jest to czas wyrażony w dziesiątych częściach sekundy, po upływie których subskrypcja utraci ważność. Po upływie tego okresu nie będą one zgodne z tą subskrypcją. Jest on również używany jako wartość w polu MDEXP w deskrypcji MQMD publikacji wysyłanych do tego subskrybenta.

Rozpoznawana jest następująca wartość specjalna:

EIULIM

Subskrypcja ma nieograniczony czas utraty ważności.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji SOALT można zmienić termin utraty ważności subskrypcji.

W przypadku powrotu z wywołania MQSUB przy użyciu opcji SORES to pole jest ustawione na pierwotne wygaśnięcie subskrypcji, a nie pozostały czas utraty ważności.

SDON (48-bajtowy łańcuch znaków)

Jest to nazwa obiektu tematu zgodnie z definicją w lokalnym menedżerze kolejek.

Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (od A do Z)
- Małe litery alfabetu (od a do z)
- Cyfry cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Użyj znaku o kodzie zero, aby wskazać koniec znaczących danych w nazwie; wartość NULL i wszystkie znaki po nim są traktowane jako znaki puste. Zastosowanie mają następujące ograniczenia:

- W systemach, w których używane jest kodowanie EBCDIC Katakana, nie można używać małych liter.

- Nazwy zawierające małe litery, ukośniki lub procenty muszą być ujęte w znaki cudzysłowu, gdy są podane w komendach. Tych znaków cudzysłowu nie należy określać dla nazw, które występują jako pola w strukturach lub jako parametry wywołań.

SDON jest używany do tworzenia pełnej nazwy tematu.

Pełna nazwa tematu może być zbudowana z dwóch różnych pól: *SDON* i *SDOS*. Szczegółowe informacje na temat sposobu użycia tych dwóch pól zawiera sekcja [“Korzystanie z łańcuchów tematów”](#) na stronie 584.

W przypadku powrotu z wywołania MQSUB za pomocą opcji SORES to pole pozostaje niezmienione.

Długość tego pola jest podana przez LNTOPN. Początkowa wartość tego pola to 48 znaków odstępu.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji SDALT nazwa subskrybowanego obiektu tematu nie może zostać zmieniona. To pole i *SDOS* można pominąć. Jeśli są one udostępniane, muszą one zostać rozstrzygane do tej samej pełnej nazwy tematu lub wywołania nie powiedzie się z produktem RC2510.

SDOPT (10-cyfrowa liczba całkowita ze znakiem)

Należy określić co najmniej jedną z następujących opcji:

- SOALT
- SORES
- SOCRT

Możliwe jest dodanie wartości. Nie należy dodawać tej samej stałej więcej niż raz. W tabeli pokazano, w jaki sposób można połączyć te opcje: zanotowane są kombinacje, które nie są poprawne; wszystkie inne kombinacje są poprawne.

Opcje dostępu lub tworzenia

Opcje dostępu i tworzenia kontrolują, czy subskrypcja jest tworzona, czy też istniejąca subskrypcja jest zwracana, czy zmieniana. Należy określić co najmniej jedną z tych opcji. W tabeli wyświetlane są poprawne kombinacje opcji dostępu lub tworzenia.

Kombinacja opcji	Uwagi
SOCRT	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje; nie powiedzie się, jeśli subskrypcja istnieje.
SORES	Wznawia istniejącą subskrypcję, jeśli subskrypcja nie istnieje.
SOCRT + SORES	Tworzy subskrypcję, jeśli nie istnieje, a następnie wznawia dopasowanie, jeśli istnieje. Przydatna kombinacja, jeśli jest używana w aplikacji, która może być uruchamiana wiele razy.
SORES + SOALT (patrz uwaga)	Wznawia istniejącą subskrypcję, modyfikując pola w celu dopasowania ich do tych określonych w tabeli MQSD. Jeśli subskrypcja nie istnieje, nie powiedzie się.
SOCRT + SOALT (patrz uwaga)	Tworzy subskrypcję, jeśli nie istnieje i wznawia dopasowanie, jeśli taka istnieje, zmieniając dowolne pola tak, aby były zgodne z tymi, które zostały określone w MQSD. Przydatna kombinacja, jeśli jest używana w aplikacji, która chce zapewnić, że jej subskrypcja znajduje się w określonym stanie przed kontynuowaniem.

Uwaga:

Opcje określające SOALT mogą również określać wartość SORES, ale ta kombinacja nie ma dodatkowego wpływu na określenie opcji SOALT w monoterapii. Wartość SOALT oznacza SORES, ponieważ wywołanie MQSUB w celu zmiany subskrypcji oznacza, że subskrypcje są również wznawiane. Odwrotność nie jest jednak prawdą, jednak: wznawianie subskrypcji nie oznacza, że ma być ona zmieniona.

SOCRT

Utwórz subskrypcję dla określonego tematu. Jeśli subskrypcja korzystała z tego samego *SDSN*, wywołanie kończy się niepowodzeniem z RC2432. Tego niepowodzenia można uniknąć, łącząc opcję SOCRT z SORES. *SDSN* nie zawsze jest konieczne. Więcej informacji na ten temat zawiera opis tego pola.

Połączenie SOCRT z SORES najpierw sprawdza, czy istnieje subskrypcja dla określonego serwera *SDSN*, a jeśli zostanie zwrócony uchwyt do tej istniejącej subskrypcji. Jeśli jednak nie istnieje subskrypcja, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól podanych w tabeli MQSD.

SOCRT można również łączyć z SOALT do podobnego efektu (patrz szczegóły na temat SOALT w dalszej części tego tematu).

SORES

Zwraca uchwyt do wcześniej istniejącej subskrypcji, która jest zgodna z subskrypcją określoną przez produkt *SDSN*. Żadne zmiany nie są wprowadzane do zgodnych atrybutów subskrypcji i są one zwracane w wyniku w strukturze MQSD. Większość treści zmaterializowanych tabel zapytań nie jest używana: używane pola to: *SDSID*, *SDVER*, *SDOPT*, *SDAID* i *SDASI* oraz *SDSN*.

Wywołanie nie powiodło się z kodem przyczyny RC2428, jeśli subskrypcja nie istnieje, zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję SOCRT z SORES. Szczegółowe informacje na temat SOCRT można znaleźć w sekcji [SOCRT](#).

ID użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika z ostatniej, pomyślnej zmiany. Jeśli używany jest produkt *SDAID*, a dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, produkt *SDAID* jest rejestrowany jako identyfikator użytkownika, który utworzył subskrypcję, a nie ID użytkownika, pod którym subskrypcja została wykonana.

Identyfikator użytkownika, który utworzył subskrypcję, jest rejestrowany jako *SDAU*, jeśli pole to jest używane, a użycie alternatywnych identyfikatorów użytkowników jest dozwolone dla tego użytkownika.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji SOAUID, a ID użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu do subskrypcji, wywołanie nie powiedzie się i kod przyczyny RC2434.

Jeśli zgodna subskrypcja istnieje i jest obecnie używana przez inną aplikację, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2429. Jeśli jest on obecnie używany przez to samo połączenie, wywołanie nie kończy się niepowodzeniem i zwracany jest uchwyt do subskrypcji.

Jeśli subskrypcja o nazwie SubName nie jest poprawną subskrypcją w celu wznowienia lub zmiany aplikacji, wywołanie zakończy się niepowodzeniem z produktem RC2523.

SORES jest implikowana przez produkt SOALT i dlatego nie jest wymagane łączenie się z tą opcją, jednak nie jest to błąd, jeśli te dwie opcje są łączone.

SOALT

Zwraca uchwyt do istniejącej subskrypcji z pełną nazwą subskrypcji zgodną z tymi podanymi w sekcji *SDSN*. Wszystkie atrybuty subskrypcji, które różnią się od atrybutów określonych w tabeli MQSD, są zmieniane w subskrypcji, chyba że zmiana jest niedozwolona dla tego atrybutu. Szczegóły znajdują się w opisie każdego atrybutu i są podsumowane w poniższej tabeli. W przypadku próby zmiany atrybutu, którego nie można zmienić, wywołanie kończy się niepowodzeniem z kodem przyczyny przedstawionym w poniższej tabeli.

Wywołanie nie powiodło się z kodem przyczyny RC2428 , jeśli subskrypcja nie istnieje, zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję SOCRT z opcją SOALT.

Połączenie SOCRT z SOALT najpierw sprawdza, czy istnieje subskrypcja dla określonej pełnej nazwy subskrypcji, a jeśli zostanie zwrócony uchwyt do tej istniejącej subskrypcji z modyfikacjami utworzonymi jako poprzednio szczegółowe; ale jeśli nie istnieje subskrypcja, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól podanych w MQSD.

ID użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika z ostatniej pomyślnej zmiany. Jeśli używany jest produkt SDAU (i użycie alternatywnych identyfikatorów użytkowników jest dozwolone dla tego użytkownika), to alternatywny identyfikator użytkownika jest rejestrowany jako identyfikator użytkownika, który utworzył subskrypcję, a nie ID użytkownika, w ramach którego subskrypcja została wykonana.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji SOAUID , a ID użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu do subskrypcji, wywołanie nie powiedzie się i kod przyczyny RC2434 .

Jeśli zgodna subskrypcja istnieje i jest obecnie używana przez inną aplikację, wywołanie zakończy się niepowodzeniem z produktem RC2429 . Jeśli jest on obecnie używany przez to samo połączenie, wywołanie nie kończy się niepowodzeniem i zwracany jest uchwyt do subskrypcji.

Jeśli subskrypcja o nazwie SubName nie jest poprawną subskrypcją w celu wznowienia lub zmiany aplikacji, wywołanie zakończy się niepowodzeniem z produktem RC2523 .

W poniższych tabelach przedstawiono atrybuty subskrypcji, które można zmieniać za pomocą komendy SOALT.

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Czy ten atrybut może zostać zmieniony za pomocą komendy SOALT?	Kod przyczyny
MQSD	Opcje trwałości	Nie	RC2509
MQSD	Opcje miejsca docelowego	Tak	Brak
MQSD	Opcje rejestracji	Tak (patrz uwaga 1)	RC2515 , jeśli próbujesz zmienić SOGRP
MQSD	Opcje publikacji	Tak (patrz uwaga 2)	Brak
MQSD	Opcje wieloznaczne	Nie	RC2510
MQSD	Inne opcje	Nie (patrz uwaga 3)	Brak
MQSD	ObjectName	Nie	RC2510
MQSD	SDAU	Nie (patrz uwaga 4)	Brak
MQSD	SDASI	Nie (patrz uwaga 4)	Brak
MQSD	SDEXP	Tak	Brak
MQSD	SDOS	Nie	RC2510
MQSD	SDSN	Nie (patrz uwaga 5)	Brak
MQSD	SDSUD	Tak	Brak

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Czy ten atrybut może zostać zmieniony za pomocą komendy SOALT?	Kod przyczyny
MQSD	ID SDCID	Tak (patrz uwaga 6)	RC2515 w przypadku subskrypcji pogrupowanej
MQSD	SDPRI	Tak	Brak
MQSD	SDACC	Tak	Brak
MQSD	SDAID	Tak	Brak
MQSD	SDSL	Nie	RC2512
MQSUB	Hobj	Tak (patrz uwaga 6)	RC2515 w przypadku subskrypcji pogrupowanej

Uwagi:

1. Nie można zmienić parametru SOGRP .
2. Nie można zmienić parametru SONEWP , ponieważ nie jest on częścią subskrypcji.
3. Te opcje nie są częścią subskrypcji
4. Ten atrybut nie jest częścią subskrypcji
5. Ten atrybut jest tożsamością modyfikowanej subskrypcji
6. Zmiana, z wyjątkiem sytuacji, gdy część pogrupowanej subskrypcji (SOGRP)

Opcje trwałości: Następujące opcje sterują sposobem, w jaki jest trwała subskrypcja. Można określić tylko jedną z tych opcji. Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji SOALT , nie można zmienić trwałości subskrypcji. Po powrocie z wywołania MQSUB za pomocą komendy SORESustawiona jest odpowiednia opcja trwałości.

SODUR

Zażądaj, aby subskrypcja tego tematu pozostała do czasu jawnego usunięcia za pomocą opcji MQCLOSE z opcją CORMSB . Jeśli ta subskrypcja nie zostanie jawnie usunięta, pozostanie ona nawet po zamknięciu tej aplikacji przez połączenie z menedżerem kolejek.

Jeśli zażądano trwałej subskrypcji do tematu, który jest zdefiniowany jako nie zezwalający na trwałe subskrypcje, wywołanie zakończy się niepowodzeniem z produktem RC2436 .

SONDUR

Żądanie subskrypcji tego tematu zostanie usunięte, gdy połączenie aplikacji z menedżerem kolejek zostanie zamknięte, jeśli jeszcze nie zostało jawnie usunięte. Parametr SONDUR jest przeciwieństwem opcji SODUR i jest zdefiniowany w dokumentacji programu pomocy. Jest to wartość domyślna, jeśli nie zostanie podana żadna wartość.

Opcje docelowe: Następujące opcje sterują miejscem docelowym, do którego wysyłane są publikacje dotyczące subskrybowanego tematu. W przypadku zmiany istniejącej subskrypcji za pomocą opcji SOALT, można zmienić miejsce docelowe używane na publikacje dotyczące subskrypcji. W przypadku powrotu z wywołania MQSUB za pomocą komendy SORESta opcja jest ustawiana, jeśli jest odpowiednia.

SOMAN

Zażądaj, aby miejsce docelowe, do którego są wysyłane publikacje, jest zarządzane przez menedżer kolejek.

Uchwyt obiektu zwrócony w produkcie HOBJ reprezentuje kolejkę zarządzaną menedżerem kolejek i jest używany z kolejnymi wywołaniami MQGET, MQCB, MQINQ lub MQCLOSE.

Uchwyt obiektu zwrócony z poprzedniego wywołania MQSUB nie może być podany w parametrze *Hobj* , jeśli nie został określony parametr *SOMAN* .

Opcje rejestracji: Następujące opcje sterują szczegółami rejestracji dokonanej w menedżerze kolejek dla tej subskrypcji. W przypadku zmiany istniejącej subskrypcji za pomocą opcji *SOALT* te opcje rejestracji mogą zostać zmienione. W przypadku powrotu z wywołania MQSUB przy użyciu komendy *SORES* ustawione są odpowiednie opcje rejestracji.

SOGRP

Ta subskrypcja jest pogrupowana z innymi subskrypcjami tego samego produktu *SDSL* przy użyciu tej samej kolejki i określając ten sam identyfikator korelacji, tak aby wszystkie publikacje dotyczące tematów, które spowodowałyby, że więcej niż jeden komunikat o publikacji były udostępniane grupie subskrypcji, ze względu na nakładający się zbiór tańcuchów tematów, które będą używane, powoduje, że tylko jeden komunikat zostanie dostarczony do kolejki. Jeśli ta opcja nie jest używana, każda subskrypcja unikalna (identyfikowana przez produkt *SDSN*) jest zgodna z kopią publikacji, która może oznaczać, że w kolejce współużytkowanej może zostać umieszczona więcej niż jedna kopia publikacji, która jest współużytkowana przez wiele subskrypcji.

Tylko najbardziej znacząca subskrypcja w grupie jest dostarczona z kopią publikacji. Najbardziej znacząca subskrypcja jest oparta na pełnej nazwie tematu aż do punktu, w którym znajduje się znak wieloznaczny. Jeśli w obrębie grupy używana jest mieszanka schematów wieloznacznych, ważna jest tylko pozycja znaku wieloznacznego. Zaleca się, aby nie łączyć różnych schematów znaków wieloznacznych w ramach grupy subskrypcji, które współużytkują tę samą kolejkę.

Podczas tworzenia nowej zgrupowanej subskrypcji musi mieć jeszcze unikalny *SDSN*, ale jeśli jest zgodny z pełną nazwą tematu istniejącej subskrypcji w grupie, wywołanie kończy się niepowodzeniem z produktem *RC2514* .

Jeśli najbardziej znacząca subskrypcja w grupie również określa *SONOLC* i jest to publikacja z tej samej aplikacji, wówczas żadna publikacja nie zostanie dostarczona do kolejki.

W przypadku zmiany subskrypcji z tą opcją, pola, które oznaczają grupowanie, *Hobj* w wywołaniu MQSUB (reprezentujące kolejkę i nazwę menedżera kolejek) i *SDCID* nie mogą być zmieniane. Próba ich zmiany spowoduje, że wywołanie zakończy się niepowodzeniem z produktem *RC2515* .

Ta opcja musi być połączona z opcją *SOSCID* z *SDCID* , która nie jest ustawiona na *CINONE*, i nie może być połączona z programem *SOMAN*.

ID SOAUID

Jeśli zostanie podana wartość *SOAUID* , tożsamość subskrybenta nie jest ograniczona do pojedynczego identyfikatora użytkownika. Dzięki temu każdy użytkownik może zmienić lub wznowić subskrypcję, gdy mają odpowiednie uprawnienia. Tylko jeden użytkownik może mieć subskrypcję w dowolnym momencie. Próba wznowienia korzystania z subskrypcji, która jest obecnie używana przez inną aplikację, powoduje niepowodzenie wywołania w produkcie *RC2429* .

Aby dodać tę opcję do istniejącej subskrypcji, wywołanie MQSUB za pomocą komendy *SOALT* musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli wywołanie MQSUB odwołuje się do istniejącej subskrypcji z ustawioną nazwą *SOAUID* , a identyfikator użytkownika różni się od oryginalnej subskrypcji, wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienia do subskrybowania tematu. Po pomyślnym zakończeniu, przyszłe publikacje tego subskrybenta są umieszczane w kolejce subskrybenta przy użyciu nowego identyfikatora użytkownika ustawionego w komunikacie publikacji.

Nie należy podawać jednocześnie wartości *SOAUID* i *SOFUID*. Jeśli nie zostanie podana żadna wartość, wartością domyślną jest *SOFUID*.

UID SOFUID

Jeśli określono wartość *SOFUID* , subskrypcja może zostać zmieniona lub wznowiona tylko przez ostatni identyfikator użytkownika w celu zmiany subskrypcji. Jeśli subskrypcja nie została zmieniona, jest to identyfikator użytkownika, który utworzył subskrypcję.

Jeśli komenda MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością SOAUID i zmienia subskrypcję za pomocą komendy SOALT w celu użycia parametru SOFUID, identyfikator użytkownika subskrypcji jest teraz stały przy użyciu tego nowego identyfikatora użytkownika. Wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienia do subskrybowania tematu.

Jeśli identyfikator użytkownika inny niż zarejestrowany jako będący właścicielem subskrypcji próbuje wznowić lub zmienić subskrypcję SOFUID, wywołanie zakończy się niepowodzeniem z produktem RC2434. Identyfikator użytkownika będącego właścicielem subskrypcji można wyświetlić za pomocą komendy **DISPLAY SBSTATUS**.

Nie należy podawać jednocześnie wartości SOAUID i SOFUID. Jeśli nie zostanie podana żadna wartość, wartością domyślną jest SOFUID.

Opcje publikacji: Następujące opcje sterują sposobem wysyłania publikacji do tego subskrybenta. W przypadku zmiany istniejącej subskrypcji za pomocą opcji SOALT te opcje publikacji można zmienić.

SONOLC

Informuje brokera o tym, że aplikacja nie chce wyświetlać żadnych własnych publikacji. Publikacje są uważane za pochodzące z tej samej aplikacji, jeśli uchwyt połączeń są takie same.

W przypadku powrotu z wywołania MQSUB za pomocą opcji SORES ta opcja jest ustawiana, jeśli jest odpowiednia.

SONEWP

Obecnie nie są wysyłane publikacje, które mają być wysyłane, gdy ta subskrypcja jest tworzona, a tylko nowe publikacje. Ta opcja ma zastosowanie tylko wtedy, gdy określona jest wartość SOCRE. Wszelkie późniejsze zmiany w subskrypcji nie wpływają na przepływ publikacji, a więc wszelkie publikacje, które zostały zachowane w temacie, zostały już wysłane do subskrybenta jako nowe publikacje.

Jeśli ta opcja zostanie podana bez komendy SOCRE, wywołanie tej opcji zakończy się niepowodzeniem z produktem RC2046. Po powrocie z wywołania MQSUB za pomocą opcji SORES ta opcja nie jest ustawiona, nawet jeśli subskrypcja została utworzona przy użyciu tej opcji.

Jeśli ta opcja nie jest używana, poprzednio zachowane komunikaty są wysyłane do podanej kolejki docelowej. Jeśli to działanie nie powiedzie się z powodu błędu (RC2525 lub RC2526), utworzenie subskrypcji nie powiedzie się.

Ta opcja nie jest poprawna w połączeniu z SOPUBR.

SOPUBR

Ustawienie tej opcji oznacza, że subskrybent żąda informacji w sposób szczególny, jeśli jest to wymagane. Menedżer kolejek nie wysyła niezamówionych komunikatów do subskrybenta. Zachowana publikacja (lub ewentualnie wiele publikacji, jeśli w temacie podano znak wieloznaczny) jest przesyłana do subskrybenta za każdym razem, gdy wywołanie MQSUBRQ jest wykonywane przy użyciu uchwytu Hsub z poprzedniego wywołania MQSUB. Żadne publikacje nie są wysyłane w wyniku wywołania MQSUB za pomocą tej opcji. W przypadku powrotu z wywołania MQSUB za pomocą opcji SORES ta opcja jest ustawiana, jeśli jest odpowiednia.

Ta opcja nie jest poprawna w połączeniu z opcją SONEWP.

Opcje wieloznaczne: Następujące opcje sterują sposobem interpretowania znaków wieloznacznych w łańcuchu udostępnionym w polu SDOS w tabeli MQSD. Można określić tylko jedną z tych opcji. W przypadku zmiany istniejącej subskrypcji za pomocą opcji SOALT te opcje wieloznaczne nie mogą zostać zmienione. W przypadku powrotu z wywołania MQSUB za pomocą komendy SORES ustawiona jest odpowiednia opcja znaku wieloznacznego.

SOWCHR

Znaki wieloznaczne działają tylko na znakach w łańcuchu tematu. Pole SOWCHR traktuje ukośnik (/) jako inny znak, który nie ma żadnego specjalnego znaczenia.

Zachowanie zdefiniowane przez parametr SOWCHR jest przedstawione w poniższej tabeli:

Znak specjalny	Zachowanie
*	Znak wieloznaczny, zero lub więcej znaków

Znak specjalny	Zachowanie
?	Znak wieloznaczny, jeden znak
%	Znak zmiany znaczenia, który pozwala na użycie znaków '*', '?' lub '%' w łańcuchu i nie jest interpretowany jako znak specjalny, na przykład '% *', '%?' lub '%% '.

Na przykład opublikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

pasuje do subskrybentów korzystających z następujących tematów:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/level2/level3/level4
```

Uwaga: Użycie znaków wieloznacznych zapewnia dokładnie znaczenie udostępnione w systemach IBM MQ V6 i WebSphere MB V6, gdy używane są komunikaty w formacie MQRFH1 w celu publikowania/subskrybowania. Zaleca się, aby nie było to używane w przypadku nowo napisanych aplikacji i jest używane tylko dla aplikacji, które wcześniej były uruchamiane względem tej wersji i nie zostały zmienione w celu użycia domyślnego zachowania ze znakami wieloznacznymi zgodnie z opisem w sekcji SOWTOP.

SOWTOP

Znaki wieloznaczne działają tylko na elementach tematu w łańcuchu tematu. Jest to zachowanie domyślne, jeśli nie zostanie wybrane żadne działanie.

Zachowanie wymagane przez komendę SOWTOP jest przedstawione w poniższej tabeli:

Znak specjalny	Zachowanie
/	Separator poziomu tematu
#	Znak wieloznaczny: wiele poziomów tematów
+	Znak wieloznaczny: pojedynczy poziom tematu

Uwaga:

Znaki '+' i '#' nie są traktowane jako znaki wieloznaczne, jeśli są one mieszane z innymi znakami (w tym samymi znakami) w obrębie poziomu tematu. W poniższym łańcuchu znaki '#' i '+' są traktowane jako zwykłe znaki.

```
level0/level1/#+/level3/level#
```

Na przykład opublikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

pasuje do subskrybentów korzystających z następujących tematów:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1+/level3/level4
```

Uwaga: To użycie znaków wieloznacznych dostarcza znaczenia udostępnionego w produkcie WebSphere Message Broker Wersja 6 podczas używania sformatowanych komunikatów MQRFH2 w celu publikowania/subskrybowania.

Inne opcje: Następujące opcje sterują sposobem wydania wywołania API, a nie subskrypcją. W przypadku powrotu z wywołania MQSUB za pomocą opcji SORES te opcje są niezmienione.

SOALTU

Pole SDAU zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności wywołania MQSUB. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy ten obiekt SDAU jest uprawniony do otwarcia obiektu przy użyciu określonych opcji dostępu, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma do tego uprawnienia.

ID SOSCID

Subskrypcja jest używana do używania identyfikatora korelacji podanego w polu *SDCID*. Jeśli ta opcja nie zostanie podana, identyfikator korelacji jest automatycznie tworzony przez menedżer kolejek w czasie subskrypcji i jest zwracany do aplikacji w polu *SDCID*. Więcej informacji na ten temat zawiera sekcja [SDCID \(24-bajtowy łańcuch bitowy\) SDCID](#).

SOSETI

Subskrypcja jest używana do używania znaczników rozliczeniowych i danych tożsamości aplikacji podanych w polach *SDACC* i *SDAID*.

Jeśli ta opcja jest określona, to ta sama kontrola autoryzacji jest przeprowadzana tak, jakby kolejka docelowa była dostępna za pomocą wywołania MQOPEN z opcją 00SETI, z wyjątkiem sytuacji, w której używana jest również opcja SOMAN, w której to przypadku nie ma kontroli autoryzacji w kolejce docelowej.

Jeśli ta opcja nie zostanie podana, publikacje wysłane do tego subskrybenta mają powiązane z nimi domyślne informacje o kontekście:

Pole w strukturze MQMD	Użyta wartość
<i>MDUID</i>	Identyfikator użytkownika powiązany z subskrypcją w momencie, w którym została wykonana subskrypcja.
<i>MDACC</i>	Określane z poziomu środowiska, jeśli jest to możliwe; jeśli nie, należy ustawić wartość ACNONE.
<i>MDAID</i>	Ustaw na puste

Ta opcja jest poprawna tylko z wartościami SOCRE i SOALT. W przypadku użycia z opcją SORES pola *SDACC* i *SDAID* są ignorowane, więc ta opcja nie ma żadnego efektu.

Jeśli subskrypcja została zmieniona bez użycia tej opcji, w której wcześniej subskrypcja dostarczyła informacje o kontekście tożsamości, dla zmienionej subskrypcji generowane są domyślne informacje o kontekście.

Jeśli subskrypcja zezwalająca na używanie różnych identyfikatorów użytkowników przy użyciu opcji SOAUID jest wznawiana przez inny identyfikator użytkownika, domyślny kontekst tożsamości jest generowany dla nowego identyfikatora użytkownika będącego właścicielem subskrypcji, a wszystkie kolejne publikacje są dostarczane wraz z nowym kontekstem tożsamości.

SOFIQ

Wywołanie MQSUB nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania.

W systemie z/OS dla aplikacji CICS lub IMS ta opcja również wymusza, że wywołanie MQSUB nie powiedzie się, jeśli połączenie jest w stanie wygaszania.

SDAU (12-bajtowy łańcuch znaków)

Jeśli zostanie podana wartość SOALTU, to pole będzie zawierać alternatywny identyfikator użytkownika, który będzie używany do sprawdzania autoryzacji subskrypcji i danych wyjściowych do kolejki docelowej (określonej w parametrze *Hobj* wywołania MQSUB), w miejsce identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli operacja się powiedzie, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji w miejscu identyfikatora użytkownika, w którym aplikacja jest aktualnie uruchomiona.

Jeśli określono parametr SOALTU i pole to jest całkowicie puste do pierwszego znaku o kodzie zero lub do końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy żadna autoryzacja użytkownika nie musi zasubskrybować tego tematu z określonymi opcjami lub kolejką docelową dla danych wyjściowych.

Jeśli parametr SOALTU nie jest określony, to pole jest ignorowane.

W przypadku powrotu z wywołania MQSUB przy użyciu komendy SORESto pole pozostaje niezmienione.

To jest pole wejściowe. Długość tego pola jest podana przez LNUID. Początkowa wartość tego pola to 12 znaków odstępu.

SDPRI (10-cyfrowa liczba całkowita ze znakiem)

Jest to wartość, która znajduje się w polu *MQPRI* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Więcej informacji na temat pola *MQPRI* w strukturze MQMD zawiera sekcja [MDPRI](#).

Wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następujących wartości specjalnych:

PRQDEF

Jeśli kolejka subskrypcji jest udostępniana w polu *Hobj* w wywołaniu MQSUB, a nie jest zarządzana, priorytet dla komunikatu jest przyjmowany z atrybutu *DefPriority* tej kolejki. Jeśli identyfikowana kolejka jest kolejką klastra lub istnieje więcej niż jedna definicja w ścieżce rozstrzygania nazw kolejek, priorytet jest określany, gdy komunikat publikacji jest umieszczany w kolejce zgodnie z opisem w sekcji [MDPRI](#).

Jeśli wywołanie MQSUB korzysta z uchwytu zarządzanego, priorytet dla komunikatu jest przyjmowany z atrybutu *DefPriority* kolejki modelowej powiązanej z subskrybowanym tematem.

PRPUB

Priorytet dla wiadomości jest priorytetem pierwotnej publikacji. Jest to początkowa wartość pola.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji SOALT można zmienić *MQPRI* wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB przy użyciu komendy SORESto pole jest ustawiane na bieżący priorytet używany w subskrypcji.

SDRO (MQCHARV)

SDRO jest długą nazwą obiektu po tym, jak menedżer kolejek rozstrzygnie nazwę udostępnioną w produkcji *SDON*.

Jeśli długa nazwa obiektu jest dostępna w produkcji *SDOS*, a w produkcji *SDON* nie jest udostępniana żadna wartość, wartość zwrócona w tym polu jest taka sama, jak wartość podana w *SDOS*.

Jeśli to pole zostanie pominięte (to znaczy SDRO.VSBufSize ma wartość zero), *SDRO* nie jest zwracany, ale długość jest zwracana w polu SDRO.VSLength. Jeśli długość jest krótsza niż pełny *SDRO*, jest ona obcinana i zwraca tyle znaków z prawej strony, co może zmieścić się w podanej długości.

Jeśli wartość *SDRO* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2520 .

SDSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury. Wartość musi być następująca:

SDSIDV

Identyfikator struktury deskryptora subskrypcji.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to *SDSIDV*

SDSL (10-cyfrowa liczba całkowita ze znakiem)

Jest to poziom powiązany z subskrypcją. Publikacje są dostarczane do tej subskrypcji tylko wtedy, gdy znajdują się w zestawie subskrypcji o największej wartości *SDSL* mniejszej lub równej wartości *PubLevel* używanej w czasie publikacji.

Wartość musi należeć do zakresu od zera do 9. Zero jest najniższym poziomem.

Wartością początkową tego pola jest 1.

Jeśli istniejąca subskrypcja zostanie zmieniona za pomocą opcji *SOALT* , nie można zmienić opcji *SDSL* .

SDSN (MQCHARV)

SDSN określa nazwę subskrypcji.

To pole jest wymagane tylko wtedy, gdy parametr *SDOPT* określa opcję *SODUR* , ale jeśli jest ona udostępniana, jest ona również używana przez menedżer kolejek dla parametru *SONDUR* . Jeśli zostanie podany, wartość *SDSN* musi być unikalna w obrębie menedżera kolejek, ponieważ jest to pole używane do identyfikowania subskrypcji.

Maksymalna długość *SDSN* wynosi 10240.

To pole służy dwóm celom. W przypadku subskrypcji *SODUR* oznacza to, że użytkownik identyfikuje subskrypcję w celu jej wznowienia po jej utworzeniu, jeśli użytkownik zamknął uchwyt w subskrypcji (za pomocą opcji *COKPSB*) lub został odłączony od menedżera kolejek. Zidentyfikowanie subskrypcji, która ma zostać usunięta po jej utworzeniu, jest wykonywane przy użyciu wywołania *MQSUB* z opcją *SORES* . Pole *SDSN* jest również wyświetlane w widoku administracyjnym subskrypcji w polu *SDSN* (*DISPLAY SBSTATUS*).

Jeśli wartość *SDSN* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, lub jeśli jest ona pominięta, gdy jest wymagana (to znaczy *SDSN*. *VCHRL* jest zerem), lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2440 .

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze *MQCHARV*.

Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji *SOALT* , nie można zmienić nazwy subskrypcji, ponieważ jest to pole używane do identyfikowania subskrypcji. Nie jest on zmieniany na wyjściu z wywołania *MQSUB* z opcją *SORES* .

SDSS (MQCHARV)

SDSS to łańcuch, który udostępnia kryteria wyboru używane podczas subskrybowania komunikatów z tematu.

To pole o zmiennej długości jest zwracane na wyjściu z wywołania *MQSUB* za pomocą opcji *SORES* , jeśli bufor jest udostępniany, a także jeśli istnieje również dodatkowa długość buforu w polu *VSBufSize*. Jeśli w wywołaniu nie zostanie podany żaden bufor, to w polu *VSLength* tabeli *MQCHARV* zwracana jest tylko długość łańcucha wyboru. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko bajty *VSBufSize* .

Jeśli wartość *SDSS* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2519 .

SDSUD (MQCHARV)

Dane udostępnione w ramach subskrypcji w tym polu są uwzględniane jako właściwość komunikatu *mq.SubUserData* każdej publikacji wystanej do tej subskrypcji.

Maksymalna długość *SDSUD* wynosi 10240.

Jeśli wartość *SDSUD* została określona niepoprawnie, zgodnie z opisem sposobu użycia struktury *MQCHARV* lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2431.

To jest pole wejściowe. Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze *MQCHARV*.

Jeśli istniejąca subskrypcja zostanie zmieniona za pomocą opcji *SOALT* , dane użytkownika subskrypcji mogą zostać zmienione.

To pole o zmiennej długości jest zwracane na wyjściu z wywołania *MQSUB* za pomocą opcji *SORES* , jeśli bufor jest udostępniony, a w produkcie *VSBuflen* istnieje dodatnia długość buforu. Jeśli w wywołaniu nie zostanie podany żaden bufor, w polu *VCHRL* obiektu *MQCHARV* zwracana jest tylko długość danych użytkownika subskrypcji. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko *VSBuflen* bajtów.

SDVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

SDVER1

Struktura deskryptora subskrypcji Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

SDVERC

Bieżąca wersja struktury deskryptora subskrypcji.

To jest zawsze pole wejściowe. Początkowa wartość pola to *SDVER1*.

Wartości początkowe

Nazwa pola	Nazwa stałej	Wartość stałej
<i>SDSID</i>	<i>SDSIDV</i>	'SD--'
<i>SDVER</i>	<i>SDVER1</i>	1
<i>SDOPT</i>	<i>SONDUR</i>	0
<i>SDON</i>	Brak	Puste
<i>SDAU</i>	Brak	Puste
<i>SDASI</i>	<i>SINONE</i>	Wartości null
<i>SDEXP</i>	<i>EIULIM</i>	-1
<i>SDOS</i>	Nazwy i wartości zdefiniowane dla tabeli <i>MQCHARV</i>	
<i>SDSN</i>	Nazwy i wartości zdefiniowane dla tabeli <i>MQCHARV</i>	
<i>SDSUD</i>	Nazwy i wartości zdefiniowane dla tabeli <i>MQCHARV</i>	

Nazwa pola	Nazwa stałej	Wartość stałej
<i>SDCID</i>	CINONE	Wartości null
<i>SDPRI</i>	PRQDEF	-3
<i>SDACC</i>	ACNONE	Wartości null
<i>SDAID</i>	Brak	Puste
<i>SDSL</i>	Brak	1
<i>SDRO</i>	Nazwy i wartości zdefiniowane w tabeli MQCHARV	
Uwaga: 1. Symbol – reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```

D*..1.....2.....3.....4.....5.....6.....7..
D* MQSD Structure
D*
D* Structure identifier
D  SDSID          1          4
D* Structure version number
D  SDVER          5          8I 0
D* Options associated with subscribing
D  SDOPT          9         12I 0
D* Object name
D  SDON          13         60
D* Alternate user identifier
D  SDAU          61         72
D* Alternate security identifier
D  SDASI         73        112
D* Expiry of Subscription
D  SDEXP        113        116I 0
D* Object Long name
D  SDOSP        117        132*
D  SDOSO        133        136I 0
D  SDOSS        137        140I 0
D  SDOSL        141        144I 0
D  SDOSC        145        148I 0
D* Subscription name
D  SDSNP        149        164*
D  SDSNO        165        168I 0
D  SDSNS        169        172I 0
D  SDSNL        173        176I 0
D  SDSNC        177        180I 0
D* Subscription User data
D  SDSUDP        181        196*
D  SDSUDO        197        200I 0
D  SDSUDS        201        204I 0
D  SDSUDL        205        208I 0
D  SDSUDC        209        212I 0
D* Correlation Id related to this subscription
D  SDCID        213        236
D* Priority set in publications
D  SDPRI        237        240I 0
D* Accounting Token set in publications
D  SDACC        241        272
D* Appl Identity Data set in publications
D  SDAID        273        304
D* Message Selector
D  SDSSP        305        320*
D  SDSSO        321        324I 0
D  SDSSS        325        328I 0
D  SDSSL        329        332I 0
D  SDSSC        333        336
D* Subscription level
D  SDSL        337        340  0
D* Resolved Long object name
D  SDROP        341        356*
D  SDR00        357        360I 0

```

D SDR0S	361	364I 0
D SDR0L	365	368I 0
D SDR0C	369	372I 0

MQSMPO-Ustawianie opcji właściwości komunikatu

Struktura produktu **MQSMPO** umożliwia aplikacjom określanie opcji, które sterują sposobem ustawiania właściwości komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym w wywołaniu **MQSETMP**.

Zestaw znaków i kodowanie: Dane w programie **MQSMPO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- [“Pola” na stronie 1247](#)
- [“Wartości początkowe” na stronie 1248](#)
- [“Deklaracja RPG” na stronie 1248](#)

Pola

Struktura MQSMPO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

SPOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje lokalizacji: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości:

SPSETF

Ustawia wartość pierwszej właściwości, która jest zgodna z podaną nazwą (lub jeśli nie istnieje), dodaje nową właściwość po wszystkich innych właściwościach z pasującą hierarchią.

SPSETC

Ustawia wartość właściwości wskazanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana za pomocą opcji IPINQF lub IPINQN.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2471.

SPSETA

Ustawia nową właściwość po właściwości wskazanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, która została ostatnio zapytana za pomocą opcji IPINQF lub IPINQO.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze utworzony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2471.

Jeśli nie jest potrzebna żadna z opisanych opcji, należy użyć następującej opcji:

BRAK

Nie określono żadnych opcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest SPSETF.

SPSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

SPSIDV

Identyfikator struktury opcji właściwości zestawu komunikatów.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **SPSIDV**.

SPVAKCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków wartości właściwości, który ma zostać ustawiony, jeśli wartość jest łańcuchem znaków.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **CSAPL**.

SPVALENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie wartości właściwości, która ma zostać ustawiona, jeśli wartość jest liczbowa.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **ENNAT**.

SPVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

SPVER1

Version-1 ustawia strukturę opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej wersji:

SPVERC

Bieżąca wersja struktury opcji właściwości komunikatu zestawu.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest **SPVER1**.

Wartości początkowe

Tabela 163. Początkowe wartości pól w MQSMPO		
Nazwa pola	Nazwa stałej	Wartość stałej
SPSID	SPSIDV	' SMP0 '
SPVER	SPVER1	1
SPOPT	BRAK	0
SPVALENC	ENNAT	Zależy od środowiska
SPVALCSI	CSAPL	-3

Deklaracja RPG

```
D* MQSMPO Structure
D*
D*
D* Structure identifier
D  SPID      1      4  INZ('SMP0')
D*
D* Structure version number
D  SPVER      5      8I 0 INZ(1)
D*
** Options that control the action of
D* MQSETMP
D  SPOPT      9      12I 0 INZ(0)
D*
D* Encoding of Value
```

D	SPVALENC	13	16I 0 INZ(273)
D*			
D*	Character set identifier of Value		
D	SPVALCSI	17	20I 0 INZ(-3)

MQSRO-Opcje żądania subskrypcji

Struktura MQSRO umożliwia aplikacji określanie opcji sterujących sposobem, w jaki jest wykonywane żądanie subskrypcji.

Przegląd

Cel: Struktura jest parametrem wejściowym/wyjściowym w wywołaniu MQSUBRQ.

Wersja: Bieżąca wersja komendy MQSRO to SRVER1.

- [“Pola” na stronie 1249](#)
- [“Wartości początkowe” na stronie 1250](#)
- [“Deklaracja RPG” na stronie 1250](#)

Pola

Struktura MQSRO zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

SRNMP (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole wyjściowe, zwrócone do aplikacji w celu wskazania liczby publikacji wysłanych do kolejki subskrypcji w wyniku tego wywołania. Mimo że ta liczba publikacji została wysłana w wyniku tego wywołania, nie ma gwarancji, że ta liczba komunikatów będzie dostępna dla aplikacji do pobrania, zwłaszcza jeśli są to komunikaty nietrwałe.

Jeśli subskrybowany temat zawiera znak wieloznaczny, może istnieć więcej niż jedna publikacja. Jeśli w łańcuchu tematu nie było żadnych znaków wieloznacznych podczas tworzenia subskrypcji reprezentowanej przez produkt *HSUB*, to w wyniku tego wywołania w większości wysyłane jest tylko jedno ogłoszenie.

SROPT (10-cyfrowa liczba całkowita ze znakiem)

Należy podać jedną z następujących opcji. Można podać tylko jedną opcję.

Inne opcje: Poniższa opcja steruje tym, co dzieje się, gdy menedżer kolejek jest wygaszany:

SRFIQ

Wywołanie MQSUBRQ nie powiodło się, jeśli menedżer kolejek znajduje się w stanie wygaszania.

Opcja domyślna: Jeśli opisana wcześniej opcja nie jest wymagana, należy użyć następującej opcji:

BRAK

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

SRNONE pomaga w dokumentacji programu. Chociaż nie jest zamierzone, aby ta opcja była używana z innymi, ponieważ jej wartość wynosi zero, nie można jej wykryć.

SRSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury. Wartość musi być następująca:

SRSIDV

Identyfikator struktury SROPT żądania subskrypcji.

To jest zawsze pole wejściowe. Wartością początkową tego pola jest SRSIDV.

SRVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

SRVER1

Struktura opcji żądania subskrypcji Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

SRVERC

Bieżąca wersja struktury opcji żądania subskrypcji.

To jest zawsze pole wejściowe. Początkowa wartość tego pola to SRVER1.

Wartości początkowe

Nazwa pola	Nazwa stałej	Wartość stałej
<i>SRSID</i>	SRSIDV	' SRO~ '
<i>SRVER</i>	SRVER1	1
<i>SROPT</i>	BRAK	0
<i>SRNMP</i>	Brak	0
Uwagi: - Symbol ~ reprezentuje pojedynczy pusty znak. - Łańcuch wartości NULL lub puste znaki są oznaczane łańcuchem pustym w języku C, a puste znaki w innych językach programowania.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D* MQSRO Structure
D*
D* Structure identifier
D  SRSID          1          4
D* Structure version number
D  SRVER          5          8I 0
D* Options that control the action of MQSUBRQ
D  SROPT          9          12I 0
D* Number of publications sent
D  SRNMP         13          16I 0
```

MQSTS-struktura raportowania statusu

Struktura MQSTS opisuje dane w strukturze statusu zwróconej przez komendę MQSTAT.

Przegląd

Zestaw znaków i kodowanie: Dane znakowe w tabeli MQSTS znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to nadawane za pomocą atrybutu menedżera kolejek *CodedCharSetId* menedżera kolejek. Dane liczbowe w tabeli MQSTS znajdują się w rodzimym kodowaniu komputera. Jest to nadawane za pomocą komendy *ENNAT*.

Użycie: Komenda MQSTAT jest używana do pobierania informacji o statusie. Te informacje są zwracane w strukturze MQSTS. Więcej informacji na temat komendy MQSTAT zawiera sekcja [“MQSTAT-pobieranie informacji o statusie”](#) na stronie 1385.

- [“Pola”](#) na stronie 1251
- [“Wartości początkowe”](#) na stronie 1254
- [“Deklaracja RPG”](#) na stronie 1254

Pola

Struktura MQSTS zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

STSCC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kod zakończenia wynikający z pierwszego błędu zgłoszonego w strukturze MQSTS.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest CCOK.

STSFC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba asynchronicznych wywołań put, które nie powiodły się.

To jest pole wyjściowe. Wartością początkową tego pola jest 0.

STSOBJN (48-bajtowy łańcuch znaków)

Jest to nazwa lokalna obiektu uczestniczonego w pierwszej awarii.

To jest pole wyjściowe. Początkowa wartość tego pola to 48 znaków odstępu.

STSOQMR (48-bajtowy łańcuch znaków)

Jest to nazwa menedżera kolejek, w którym zdefiniowany jest obiekt *STSOBJN*. Nazwa, która jest całkowicie pusta, do pierwszego znaku o wartości NULL lub do końca pola oznacza menedżer kolejek, z którym połączona jest aplikacja (lokalny menedżer kolejek).

To jest pole wyjściowe. Początkowa wartość tego pola to 48 znaków odstępu.

STS00 (10-cyfrowa liczba całkowita ze znakiem)

STS00 używany do otwierania zgłaszanego obiektu. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Wartość STS00 zależy od wartości parametru MQSTAT STYPE.

STATAPT

Zero.

STATREC

Zero.

STATRER

STS00, który był używany w przypadku wystąpienia błędu. Przyczyna niepowodzenia jest zgłaszana w polach *STSCC* i *STSRC* w strukturze MQSTS.

STS00 jest polem wyjściowym. Jego początkowa wartość wynosi zero.

STSOS (MQCHARV)

Długa nazwa obiektu, dla którego zgłaszany jest błąd obiektu. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

STSOS to pole MQCHARV o maksymalnej długości 10240. Opis sposobu korzystania z struktury MQCHARV zawiera sekcja [MQCHARV](#).

Interpretacja wartości STSOS zależy od wartości parametru MQSTAT STYPE.

STATAPT

Jest to długa nazwa obiektu dla kolejki lub tematu używanego w operacji MQPUT, która nie powiodła się.

STATREC

Łańcuch o zerowej długości

STATRER

Jest to długa nazwa obiektu, który spowodował niepowodzenie ponownego nawiązania połączenia.

STSOS jest polem wyjściowym. Jego początkowa wartość to łańcuch o zerowej długości.

STSOT (10-cyfrowa liczba całkowita ze znakiem)

Typ obiektu, który jest nazwany w programie *ObjectName*. Dozwolone są następujące wartości:

OTALSQ

Kolejka aliasowa.

OTLOCQ

Kolejka lokalna.

OTMODQ

Kolejka modelowa.

OTQ

do kolejki błędów.

OTREMQ

Kolejka zdalna.

OTTOP

.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest OTQ.

STSRC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kod przyczyny wynikający z pierwszego błędu zgłoszonego w strukturze MQSTS.

To jest zawsze pole wyjściowe. Wartością początkową tego pola jest RCNONE.

STSRBJN (48-bajtowy łańcuch znaków)

Jest to nazwa kolejki docelowej o nazwie określonej w *STSRBJN*, po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa kolejki, która istnieje w menedżerze kolejek identyfikowanego przez produkt *STSRQMGR*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *STSRBJN* jest pusty:

- Temat
- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.

To jest pole wyjściowe. Początkowa wartość tego pola to 48 znaków odstępu.

STSRQMGR (48-bajtowy łańcuch znaków)

Jest to nazwa docelowego menedżera kolejek po tłumaczeniu nazwy przez lokalny menedżer kolejek. Zwrócona nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *STSRBJN*. *STSRQMGR* może być nazwą lokalnego menedżera kolejek.

Jeśli *STSRBJN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, *STSRQMGR* jest nazwą grupy współużytkowania kolejki. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, *STSRBJN* może być nazwą grupy współużytkowania kolejki lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejki (rodzaj zwróconej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą dla przeglądania, wejścia lub wyjścia (lub dowolnej kombinacji). Jeśli obiekt otwarty jest dowolnym z poniższych obiektów, *STSRQMGR* jest pusty:

- Temat

- Kolejka, ale nie została otwarta do przeglądania, wprowadzania danych lub danych wyjściowych.
- Kolejka kłakra o podanej nazwie OOBNDN (lub z wartością OOBNDQ w przypadku, gdy atrybut kolejki *DefBind* ma wartość OOBNDN)

To jest pole wyjściowe. Początkowa wartość tego pola to 48 znaków odstępu.

STSSC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba wywołań put asynchronicznych, które powiodły się.

To jest pole wyjściowe. Wartością początkową tego pola jest 0.

STSSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury. Wartość musi być następująca:

ID STSSID

Identyfikator struktury raportowania statusu.

Wartością początkową tego pola jest STSSID.

STSSO (10-cyfrowa liczba całkowita ze znakiem)

STSSO używana do otwierania uszkodzonej subskrypcji. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

Interpretacja wartości STSSO zależy od wartości parametru MQSTAT STYPE .

STATAPT

Zero.

STATREC

Zero.

STATRER

STSSO , który był używany w przypadku wystąpienia błędu. Przyczyna niepowodzenia jest zgłaszana w polach *STSCC* i *STSRC* w strukturze MQSTS . Jeśli niepowodzenie nie jest powiązane z subskrypcją tematu, zwrócona wartość wynosi zero.

STSSO jest polem wyjściowym. Jego początkowa wartość wynosi zero.

STSSUN (MQCHARV)

Nazwa niesprawnej subskrypcji. Dostępne tylko w wersji 2 produktu MQSTS lub nowszej.

STSSUN to pole MQCHARV z maksymalną długością 10240. Opis sposobu korzystania z struktury MQCHARV zawiera sekcja [MQCHARV](#) .

Interpretacja wartości STSSUN zależy od wartości parametru MQSTAT STYPE .

STATAPT

Łańcuch o zerowej długości.

STATREC

Łańcuch o zerowej długości.

STATRER

Nazwa subskrypcji, która spowodowała niepowodzenie ponownego nawiązania połączenia. Jeśli żadna nazwa subskrypcji nie jest dostępna lub niepowodzenie nie jest powiązane z subskrypcją, jest to łańcuch o zerowej długości.

STSSUN jest polem wyjściowym. Jego początkowa wartość to łańcuch o zerowej długości.

STSVR (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

STSVR1

Numer wersji struktury raportowania statusu.

Następująca stała określa numer wersji bieżącej wersji:

STSVRC

Bieżąca wersja struktury raportowania statusu.

Początkowa wartość tego pola to STSVR1.

STSWC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba asynchronicznych wywołań put, które zakończyły się ostrzeżeniem.

To jest pole wyjściowe. Wartością początkową tego pola jest 0.

Wartości początkowe

Tabela 164. Początkowe wartości pól w MQSTS		
Nazwa pola	Nazwa stałej	Wartość stałej
STSSID	STSID	
STSVER	STSVRC	STSVR1
STSCC	CCOK	0
STSRC	RCBRAK	0
STSSC	Brak	0
STSWC	Brak	0
STSFC	Brak	0
STSOT	Brak	0
STSOBJN	Brak	Puste
STSOQMGR	Brak	Puste
STSR OBJN	Brak	Puste
STSRQMGR	Brak	Puste
STSOS	Nazwy i wartości zdefiniowane dla tabeli MQCHARV	
STSSUN	Nazwy i wartości zdefiniowane dla tabeli MQCHARV	
STS00	Brak	0
STSS0	Brak	0

Deklaracja RPG

```

D*.1.....2.....3.....4.....5.....6.....7..
D* MQSTS Structure
D*
D* Structure identifier
D STSSID          1      4
D* Structure version number
D STSVER          5      8I 0
D* Completion code
D STSCC           9      12I 0
D* Reason code
D STSRC          13      16I 0
D* Success count

```

D	STSSC	17	20I 0
D*	Warning count		
D	STSWC	21	24I 0
D*	Failure count		
D	STSFC	25	28I 0
D*	Object type		
D	STSOT	29	32I 0
D*	Object name		
D	STSOBJN	33	80
D*	Object queue manager		
D	STSOQMGR	81	128
D*	Resolved object name		
D	STSR OBJN	129	176
D*	Resolved object queue manager name		
D	STSRQMGR	177	224
D*	Ver:1 **		
D*	Failing object long name		
D*	Address of variable length string		
D	STSOSCHRP	225	240*
D*	Offset of variable length string		
D	STSOSCHRO	241	244I 0
D*	Size of buffer		
D	STSOSVSBS	245	248I 0
D*	Length of variable length string		
D	STSOSCHRL	249	252I 0
D*	CCSID of variable length string		
D	STSOSCHRC	253	256I 0
D*	Failing subscription name		
D*	Address of variable length string		
D	STSSUNCHRP	257	272*
D*	Offset of variable length string		
D	STSSUNCHRO	273	276I 0
D*	Size of buffer		
D	STSSUNVSBS	277	280I 0
D*	Length of variable length string		
D	STSSUNCHRL	281	284I 0
D*	CCSID of variable length string		
D	STSSUNCHRC	285	288I 0
D*	Failing open options		
D	STS00	289	292I 0
D*	Failing subscription options		
D	STSS0	293	296I 0
D*	Ver:2 **		

MQTM-komunikat wyzwalacza

Struktura MQTM opisuje dane w komunikacie wyzwalacza, który jest wysyłany przez menedżer kolejek do aplikacji monitora wyzwalacza, gdy wystąpi zdarzenie wyzwalające dla kolejki.

Przegląd

Przeznaczenie: ta struktura jest częścią interfejsu programu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska produktu IBM MQ .

Nazwa formatu: FMTM.

Zestaw znaków i kodowanie: Dane znakowe w tabeli MQTM znajdują się w zestawie znaków menedżera kolejek, który generuje tabelę MQTM. Dane liczbowe w tabeli MQTM znajdują się w kodowaniu maszyny menedżera kolejek, który generuje program MQTM.

Zestaw znaków i kodowanie tabeli MQTM są podane w polach *MDCSI* i *MDENC* w:

- MQMD (jeśli struktura MQTM znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQTM (wszystkie inne obserwacje).

Użycie: Aplikacja monitorującego wyzwalacza może wymagać przekazania niektórych lub wszystkich informacji w komunikacie wyzwalacza do aplikacji, która jest uruchamiana przez aplikację wyzwalacza-monitor. Informacje, które mogą być potrzebne w uruchomionej aplikacji, obejmują produkty *TMQN*, *TMTDi* *TMUD*. Aplikacja wyzwalanie-monitor może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone przez środowisko i wygodne dla uruchomionej aplikacji. Informacje na temat MQTMC2 można znaleźć w sekcji [“MQTMC2 -komunikat wyzwalacza 2 \(format znakowy\)”](#) na stronie 1259.

- W systemie IBM i aplikacja wyzwalacza-monitor dostarczona z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Więcej informacji na temat wyzwalaczy zawiera sekcja [Wymagania wstępne dla wyzwalania](#).

- [“MQMD dla komunikatu wyzwalacza” na stronie 1256](#)
- [“Pola” na stronie 1257](#)
- [“Wartości początkowe” na stronie 1259](#)
- [“Deklaracja RPG” na stronie 1259](#)

MQMD dla komunikatu wyzwalacza

MQMD dla komunikatu wyzwalacza: Pola w strukturze MQMD komunikatu wyzwalacza generowanego przez menedżer kolejek są ustawiane w następujący sposób:

Pole w strukturze MQMD	Użyta wartość
<i>MDSID</i>	MDSIDV
<i>MDVER</i>	MDVER1
<i>MDREP</i>	RONONE
<i>MDMT</i>	MTDGRM
<i>MDEXP</i>	EIULIM
<i>MDFB</i>	FBNONE
<i>MDENC</i>	ENNAT
<i>MDCSI</i>	Atrybut <i>CodedCharSetId</i> menedżera kolejek
<i>MDFMT</i>	FMTM
<i>MDPRI</i>	Atrybut <i>DefPriority</i> kolejki inicjuj
<i>MDPER</i>	PENPER
<i>MDMID</i>	Unikalna wartość
<i>MDCID</i>	CINONE
<i>MDBOC</i>	0
<i>MDRQ</i>	Puste
<i>MDRM</i>	Nazwa menedżera kolejek.
<i>MDUID</i>	Puste
<i>MDACC</i>	ACNONE
<i>MDAID</i>	Puste
<i>MDPAT</i>	ATQM lub jako odpowiedni dla agenta kanału komunikatów
<i>MDPAN</i>	Pierwsze 28 bajtów nazwy menedżera kolejek
<i>MDPD</i>	Data wysłania komunikatu wyzwalacza
<i>MDPT</i>	Czas wysłania komunikatu wyzwalacza
<i>MDAOD</i>	Puste

W celu ustawienia podobnych wartości zaleca się stosowanie aplikacji generującej komunikat wyzwalacza, z wyjątkiem następujących:

- Pole *MDPRI* można ustawić na wartość PRQDEF (menedżer kolejek zmieni to ustawienie na priorytet domyślny dla kolejki inicjuj, gdy komunikat jest umieszczany).

- Pole *MDRM* można ustawić na puste (menedżer kolejek zmieni to nazwę na nazwę lokalnego menedżera kolejek po umieszczeniu w nim komunikacie).
- Pola kontekstu powinny być ustawione odpowiednio dla aplikacji.

Pola

Struktura MQTM zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

TMAI (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Jest to łańcuch znaków identyfikujący aplikację, która ma być uruchomiona, i jest używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *AppLId* obiektu procesu identyfikowanego przez pole *TMPN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 1425](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

Znaczenie *TMAI* jest określone przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez produkt IBM MQ wymaga, aby *TMAI* była nazwą programu wykonywalnego.

Długość tego pola jest podana przez LNPROA. Początkowa wartość tego pola to 256 znaków odstępu.

TMAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji.

Identyfikuje on rodzaj programu do uruchomienia i jest używany przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *AppLType* obiektu procesu identyfikowanego przez pole *TMPN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 1425](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

TMAT może mieć jedną z następujących wartości standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu ATUFST przez ATULST:

równaCICS

CICS.

ATVSE

CICS/VSE.

AT400

Aplikacja IBM i.

ATUFST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Wartością początkową tego pola jest 0.

TMED (128-bajtowy łańcuch znaków)

Dane środowiska.

Jest to łańcuch znaków zawierający informacje dotyczące środowiska dotyczące aplikacji, która ma być uruchomiona, i jest używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *EnvData* obiektu procesu identyfikowanego przez pole *TMPN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 1425](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest podana przez LNPROE. Początkowa wartość tego pola to 128 znaków odstępu.

TMPN (48-bajtowy łańcuch znaków)

Nazwa obiektu procesu.

Jest to nazwa obiektu procesu menedżera kolejek określonego dla kolejki wyzwalanej, która może być używana przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *ProcessName* kolejki identyfikowanej przez pole *TMQN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla kolejek” na stronie 1395](#).

Nazwy, które są krótsze od zdefiniowanej długości pola, są zawsze dopełniane do prawej strony odstępami; nie są one kończone przedwcześnie znakiem o kodzie zero.

Długość tego pola jest podana przez LNPRON. Początkowa wartość tego pola to 48 znaków odstępu.

TMQN (48-bajtowy łańcuch znaków)

Nazwa wyzwalanej kolejki.

Jest to nazwa kolejki, dla której wystąpiło zdarzenie wyzwalające, i jest używana przez aplikację uruchomioną przez aplikację monitorującego wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *QName* wyzwalanej kolejki. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla kolejek” na stronie 1395](#).

Nazwy, które są krótsze od zdefiniowanej długości pola, są dopełniane do prawej strony odstępami; nie są one kończone przedwcześnie znakiem o kodzie zero.

Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

TMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

TMSIDV

Identyfikator struktury komunikatu wyzwalacza.

Wartością początkową tego pola jest TMSIDV.

TMTD (64-bajtowy łańcuch znaków)

Dane wyzwalacza.

Jest to dane w formacie wolnoformatowym do użycia przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *TriggerData* kolejki identyfikowanej przez pole *TMQN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty dla kolejek” na stronie 1395](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest podana przez LNTRGD. Początkowa wartość tego pola to 64 znaki puste.

TMUD (128-bajtowy łańcuch znaków)

Dane użytkownika.

Jest to łańcuch znaków zawierający informacje o użytkowniku, które są istotne dla aplikacji, która ma być uruchomiona, i jest używany przez aplikację wyzwalacza-monitor, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole za pomocą wartości atrybutu *UserData* obiektu procesu identyfikowanego przez pole *TMPN*. Szczegółowe informacje na temat tego atrybutu zawiera sekcja [“Atrybuty definicji procesów” na stronie 1425](#). Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest podana przez LNPROU. Początkowa wartość tego pola to 128 znaków odstępu.

TMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

TMVER1

Numer wersji struktury komunikatu wyzwalacza.

Następująca stała określa numer wersji bieżącej wersji:

TMVERC

Bieżąca wersja struktury komunikatu wyzwalacza.

Początkowa wartość tego pola to TMVER1.

Wartości początkowe

Tabela 165. Początkowe wartości pól w MQTM		
Nazwa pola	Nazwa stałej	Wartość stałej
TMSID	TMSIDV	'TM¬¬'
TMVER	TMVER1	1
TMQN	Brak	Puste
TMPN	Brak	Puste
TMTD	Brak	Puste
TMAT	Brak	0
TMAI	Brak	Puste
TMED	Brak	Puste
TMUD	Brak	Puste
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQTM Structure
D*
D* Structure identifier
D  TMSID          1      4    INZ('TM ')
D* Structure version number
D  TMVER          5      8I 0 INZ(1)
D* Name of triggered queue
D  TMQN          9      56    INZ
D* Name of process object
D  TMPN         57     104    INZ
D* Trigger data
D  TMTD        105     168    INZ
D* Application type
D  TMAT        169     172I 0 INZ(0)
D* Application identifier
D  TMAI        173     428    INZ
D* Environment data
D  TMED        429     556    INZ
D* User data
D  TMUD        557     684    INZ
```

MQTMC2 -komunikat wyzwalacza 2 (format znakowy)

Gdy aplikacja wyzwalana wyzwalaczem pobiera komunikat wyzwalacza (MQTM) z kolejki inicjuj, monitor wyzwalacza może wymagać przekazania niektórych lub wszystkich informacji w komunikacie wyzwalacza do aplikacji, która jest uruchamiana przez monitor wyzwalacza.

Przegląd

Przeznaczenie: Informacje, które mogą być wymagane przez aplikację uruchamiającą, obejmują produkty *TC2QN*, *TC2TDi* *TC2UD*. Aplikacja monitorującego wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2, w zależności od tego, co jest dozwolone przez środowisko i wygodne dla uruchomionej aplikacji.

Ta struktura jest częścią interfejsu programu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska produktu IBM MQ.

Zestaw znaków i kodowanie: Dane znakowe w MQTMC2 znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to dane nadawane przez atrybut menedżera kolejek produktu *CodedCharSetId*.

Użycie: Struktura MQTMC2 jest taka jak format struktury MQTM. Różnica polega na tym, że pola nieznakowe w MQTM są zmieniane w MQTMC2 na pola znakowe o tej samej długości, a nazwa menedżera kolejek jest dodawana na końcu struktury.

- W systemie IBM i aplikacja monitora wyzwalacza dostarczona z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.
- [“Pola” na stronie 1260](#)
- [“Wartości początkowe” na stronie 1261](#)
- [“Deklaracja RPG” na stronie 1261](#)

Pola

Struktura MQTMC2 zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

TC2AI (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Zapoznaj się z polem *TMAI* w strukturze MQTM.

TC2AT (4-bajtowy łańcuch znaków)

Typ aplikacji.

To pole zawsze zawiera spacje, niezależnie od wartości w polu *TMAT* w strukturze MQTM oryginalnego komunikatu wyzwalacza.

TC2ED (128-bajtowy łańcuch znaków)

Dane środowiska.

Zapoznaj się z polem *TMED* w strukturze MQTM.

TC2PN (48-bajtowy łańcuch znaków)

Nazwa obiektu procesu.

Zapoznaj się z polem *TMPN* w strukturze MQTM.

TC2QMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, w którym wystąpiło zdarzenie wyzwalające.

TC2QN (48-bajtowy łańcuch znaków)

Nazwa wyzwalanej kolejki.

Zapoznaj się z polem *TMQN* w strukturze MQTM.

TC2SID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

TCSIDV

Identyfikator struktury komunikatu wyzwalacza (format znakowy).

TC2TD (64-bajtowy łańcuch znaków)

Dane wyzwalacza.

Zapoznaj się z polem *TMTD* w strukturze MQTM.

TC2UD (128-bajtowy łańcuch znaków)

Dane użytkownika.

Zapoznaj się z polem *TMUD* w strukturze MQTM.

TC2VER (4-bajtowy łańcuch znaków)

Numer wersji struktury.

Wartość musi być następująca:

TCVER2

Struktura komunikatu wyzwalacza wersji 2 (format znakowy).

Następująca stała określa numer wersji bieżącej wersji:

TCVERC

Bieżąca wersja struktury komunikatu wyzwalacza (format znakowy).

Wartości początkowe

Tabela 166. Początkowe wartości pól w MQTMC2		
Nazwa pola	Nazwa stałej	Wartość stałej
TC2SID	TCSIDV	'TMC¬'
TC2VER	TCVER2	'¬¬2'
TC2QN	Brak	Puste
TC2PN	Brak	Puste
TC2TD	Brak	Puste
TC2AT	Brak	Puste
TC2AI	Brak	Puste
TC2ED	Brak	Puste
TC2UD	Brak	Puste
TC2QMN	Brak	Puste
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQTMC2 Structure
D*
D* Structure identifier
D TC2SID 1 4
D* Structure version number
D TC2VER 5 8
```

D*	Name of triggered queue		
D	TC2QN	9	56
D*	Name of process object		
D	TC2PN	57	104
D*	Trigger data		
D	TC2TD	105	168
D*	Application type		
D	TC2AT	169	172
D*	Application identifier		
D	TC2AI	173	428
D*	Environment data		
D	TC2ED	429	556
D*	User data		
D	TC2UD	557	684
D*	Queue manager name		
D	TC2QMN	685	732

MQWIH-nagłówek informacji o pracy

Struktura MQWIH opisuje informacje, które muszą być obecne na początku komunikatu, który ma być obsługiwany przez menedżer obciążenia produktu z/OS .

Przegląd

Nazwa formatu: FMWIH.

Zestaw znaków i kodowanie: pola w strukturze MQWIH znajdują się w zestawie znaków i kodowaniu podanym w polach *MDCSI* i *MDENC* w strukturze nagłówka poprzedzającym MQWIH lub przez te pola w strukturze MQMD, jeśli wartość MQWIH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach kolejek.

Użycie: Jeśli komunikat ma być przetworzony przez menedżera obciążenia produktu z/OS , komunikat musi rozpoczynać się od struktury MQWIH.

- [“Pola” na stronie 1262](#)
- [“Wartości początkowe” na stronie 1264](#)
- [“Deklaracja RPG” na stronie 1264](#)

Pola

Struktura MQWIH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

WICSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych, który jest zgodny z MQWIH.

Określa identyfikator zestawu znaków dla danych, które są zgodne ze strukturą MQWIH. Nie ma on zastosowania do danych znakowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT

Dziedziczy identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *po* tej strukturze są w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysłanej w komunikacie na rzeczywisty identyfikator zestawu znaków w strukturze. Jeśli błąd nie zostanie zgłoszony, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Wartość CSINHT nie może być używana, jeśli wartością pola *MDPAT* w deskrypcie MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

WIENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne danych, które są zgodne z MQWIH.

Określa kodowanie numeryczne danych, które są zgodne ze strukturą MQWIH; nie ma zastosowania do danych liczbowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

WIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi

Wartość musi być następująca:

WINONE

Brak flag.

Wartością początkową tego pola jest WINONE.

WIFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych, które są następujące: MQWIH.

Określa nazwę formatu danych, które są zgodne ze strukturą MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić to pole na wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Długość tego pola jest podana przez LNFMT. Wartością początkową tego pola jest FMNONE.

WILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQWIH.

Wartość musi być następująca:

WILEN1

Długość struktury nagłówka informacji o pracy w wersji version-1 .

Następująca stała określa długość bieżącej wersji:

WILENC

Długość bieżącej wersji struktury nagłówka informacji o pracy.

Początkowa wartość tego pola to WILEN1.

WIRSV (32-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Musi być puste.

WISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

WISIDV

Identyfikator struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest WISIDV.

WISNM (32-bajtowy łańcuch znaków)

Nazwa usługi.

Jest to nazwa usługi, która ma przetworzyć komunikat.

Długość tego pola jest podana przez LNSVNM. Początkowa wartość tego pola to 32 znaki puste.

WISST (8-bajtowy łańcuch znaków)

Nazwa kroku usługi.

Jest to nazwa kroku *WISNM*, do którego odnosi się komunikat.

Długość tego pola jest podana przez LNSVST. Początkowa wartość tego pola to 8 znaków odstępu.

WITOK (16-bajtowy łańcuch bitowy)

Token komunikatu.

Jest to znacznik komunikatu, który jednoznacznie identyfikuje komunikat.

W przypadku wywołań MQPUT i MQPUT1 to pole jest ignorowane. Długość tego pola jest podana przez LNMTOK. Wartością początkową tego pola jest MTKNON.

WIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

WIVER1

Struktura nagłówka informacji o pracy Version-1.

Następująca stała określa numer wersji bieżącej wersji:

WIVERC

Bieżąca wersja struktury nagłówka informacji o pracy.

Początkowa wartość tego pola to WIVER1.

Wartości początkowe

Tabela 167. Początkowe wartości pól w tabeli MQWIH		
Nazwa pola	Nazwa stałej	Wartość stałej
WISID	WISIDV	'WIH¬'
WIVER	WIVER1	1
WILEN	WILEN1	120
WIENC	Brak	0
WICSI	CSUNDF	0
WIFMT	FMNONE	Puste
WIFLG	WINONE	0
WISNM	Brak	Puste
WISST	Brak	Puste
WITOK	MTKNON	Wartości null
WIRSV	Brak	Puste
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

D*..1.....2.....3.....4.....5.....6.....7..
D*

```

D* MQWIH Structure
D*
D* Structure identifier
D WISID 1 4 INZ('WIH ')
D* Structure version number
D WIVER 5 8I 0 INZ(1)
D* Length of MQWIH structure
D WILEN 9 12I 0 INZ(120)
D* Numeric encoding of data that followsMQWIH
D WIENC 13 16I 0 INZ(0)
D* Character-set identifier of data thatfollows MQWIH
D WICSI 17 20I 0 INZ(0)
D* Format name of data that followsMQWIH
D WIFMT 21 28 INZ(' ')
D* Flags
D WIFLG 29 32I 0 INZ(0)
D* Service name
D WISNM 33 64 INZ
D* Service step name
D WISST 65 72 INZ
D* Message token
D WITOK 73 88 INZ(X'0000000000000000-
D 0000000000000000')
D* Reserved
D WIRSV 89 120 INZ

```

MQXQH-nagłówek kolejki transmisji

Struktura MQXQH opisuje informacje, które są poprzedzane danymi komunikatów aplikacji, gdy znajdują się one w kolejkach transmisji.

Przegląd

Przeznaczenie: Kolejka transmisji jest specjalnym typem kolejki lokalnej, która tymczasowo przechowuje komunikaty przeznaczone dla kolejek zdalnych (czyli jest przeznaczone dla kolejek, które nie należą do lokalnego menedżera kolejek). Kolejka transmisji jest oznaczana za pomocą atrybutu kolejki *Usage* o wartości USTRAN.

Nazwa formatu: FMXQH.

Zestaw znaków i kodowanie: Dane w tabeli MQXQH muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek podanego przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQXQH muszą być ustawione w polach *MDCSI* i *MDENC* w:

- Osobny deskryptor MQMD (jeśli struktura MQXQH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka, która poprzedza strukturę MQXQH (wszystkie inne obserwacje).

Użycie: Komunikat, który znajduje się w kolejce transmisji, ma *dwa* deskryptory komunikatów:

- Jeden deskryptor komunikatu jest przechowywany oddzielnie od danych komunikatu, jest nazywany *odrębnym deskryptorem komunikatu* jest generowany przez menedżer kolejek, gdy komunikat jest umieszczany w kolejce transmisji. Niektóre pola w oddzielnym deskryptorze komunikatu są kopiowane z deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Osobny deskryptor komunikatu jest to ten, który jest zwracany do aplikacji w parametrze *MSGDSC* wywołania MQGET, gdy komunikat jest usuwany z kolejki transmisji.

- Drugi deskryptor komunikatu jest przechowywany w strukturze MQXQH jako część danych komunikatu. Jest to nazywane *osadzonym deskryptorem komunikatu* jest kopią deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 (z niewielkimi zmianami).

Osadzony deskryptor komunikatu jest zawsze MQMD w wersji version-1 . Jeśli komunikat umieszczony przez aplikację ma wartości inne niż domyślne dla co najmniej jednej z pól version-2 w strukturze MQMD, struktura MQMDE jest zgodna z tabelą MQXQH, po czym następuje po kolei dane komunikatu aplikacji (jeśli istnieją). MQMDE:

- Wygenerowane przez menedżer kolejek (jeśli aplikacja używa deskryptora MQMD version-2 w celu umieszczenia komunikatu), lub

- Jest już obecny na początku danych komunikatu aplikacji (jeśli aplikacja używa deskryptora MQMD w wersji version-1 do umieszczenia komunikatu).

Osadzony deskryptor komunikatu jest tym, który jest zwracany do aplikacji w parametrze *MSGDSC* wywołania MQGET, gdy komunikat jest usuwany z końcowej kolejki docelowej.

- [“Pola w oddzielnym deskrypcorze komunikatu” na stronie 1266](#)
- [“Pola w osadzonym deskrypcorze komunikatu” na stronie 1267](#)
- [“Umieszczanie komunikatów w kolejkach zdalnych” na stronie 1267](#)
- [“Umieszczanie komunikatów bezpośrednio w kolejkach transmisji” na stronie 1268](#)
- [“Pobieranie komunikatów z kolejek transmisji” na stronie 1268](#)
- [“Pola” na stronie 1268](#)
- [“Wartości początkowe” na stronie 1269](#)
- [“Deklaracja RPG” na stronie 1269](#)

Pola w oddzielnym deskrypcorze komunikatu

Pola w oddzielnym deskrypcorze komunikatu są ustawiane przez menedżer kolejek zgodnie z poniższą listą. Jeśli menedżer kolejek nie obsługuje deskryptora MQMD z wersji version-2 , nie jest używana funkcja MQMD w wersji version-1 bez utraty funkcji.

Pole w oddzielnej tabeli MQMD	Użyta wartość
<i>MDSID</i>	MDSIDV
<i>MDVER</i>	MDVER2
<i>MDREP</i>	Skopiowano z osadzonego deskryptora komunikatu, ale z bitami określonymi przez parametr ROAUXM ustawionym na zero. (uniemożliwia to wygenerowanie komunikatu raportu COA lub COD, gdy komunikat jest umieszczany w kolejce transmisji lub usuwany z kolejki transmisji).
<i>MDMT</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDEXP</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDFB</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDENC</i>	ENNAT
<i>MDCSI</i>	Atrybut <i>CodedCharSetId</i> menedżera kolejek.
<i>MDFMT</i>	FMXQH
<i>MDPRI</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDPER</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDMID</i>	Nowa wartość jest generowana przez menedżer kolejek. Ten identyfikator komunikatu różni się od <i>MDMID</i> , który menedżer kolejek mógł wygenerować dla osadzonego deskryptora komunikatu (patrz opis wcześniej).
<i>MDCID</i>	<i>MDMID</i> z osadzonego deskryptora komunikatu.
<i>MDBOC</i>	0
<i>MDRQ</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDRM</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDUID</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDACC</i>	Skopiowano z osadzonego deskryptora komunikatu.

Pole w oddzielnej tabeli MQMD	Użyta wartość
<i>MDAID</i>	Skopiowano z osadzonego deskryptora komunikatu.
<i>MDPAT</i>	ATQM
<i>MDPAN</i>	Pierwsze 28 bajtów nazwy menedżera kolejek.
<i>MDPD</i>	Data umieszczenia komunikatu w kolejce transmisji.
<i>MDPT</i>	Czas umieszczenia komunikatu w kolejce transmisji.
<i>MDAOD</i>	Puste
<i>MDGID</i>	GINONE
<i>MDSEQ</i>	1
<i>MDOFF</i>	0
<i>MDMFL</i>	MFBRK
<i>MDOLN</i>	OLUNDF

Pola w osadzonym deskrypcorze komunikatu

Pola w osadzonym deskrypcorze komunikatu mają te same wartości, co wartości w parametrze *MSGDSC* wywołania MQPUT lub MQPUT1, z wyjątkiem następujących:

- Pole *MDVER* zawsze ma wartość *MDVER1*.
- Jeśli pole *MDPRI* ma wartość *PRQDEF*, to jest zastępowane wartością atrybutu *DefPriority* kolejki.
- Jeśli pole *MDPER* ma wartość *PEQDEF*, to jest zastępowane wartością atrybutu *DefPersistence* kolejki.
- Jeśli pole *MDMID* ma wartość *MINONE*, lub podano opcję *PMNMID* lub komunikat jest komunikatem listy dystrybucyjnej, *MDMID* jest zastępowany nowym identyfikatorem komunikatu wygenerowanym przez menedżer kolejek.

Gdy komunikat z listą dystrybucyjną jest dzielony na mniejsze komunikaty listy dystrybucyjnej umieszczone w różnych kolejkach transmisji, pole *MDMID* w każdym nowym osadzonym deskrypcorze komunikatów jest takie samo, jak w oryginalnym komunikacie listy dystrybucyjnej.

- Jeśli została określona opcja *PMNCID*, *MDCID* jest zastępowana przez nowy identyfikator korelacji wygenerowany przez menedżer kolejek.
- Pola kontekstu są ustawiane zgodnie z opcjami *PM** określonymi w parametrze *PMO*. Pola kontekstu są następujące:
 - *MDACC*
 - *MDAID*
 - *MDAOD*
 - *MDPAN*
 - *MDPAT*
 - *MDPD*
 - *MDPT*
 - *MDUID*
- Pola version-2 (jeśli były obecne) są usuwane z deskryptora MQMD, a następnie przenoszone do struktury MQMDE, jeśli co najmniej jedna z pól version-2 ma wartość niedomyślną.

Umieszczanie komunikatów w kolejkach zdalnych

: Gdy aplikacja umieszcza komunikat w kolejce zdalnej (poprzez podanie nazwy kolejki zdalnej bezpośrednio lub przy użyciu lokalnej definicji kolejki zdalnej), lokalny menedżer kolejek:

- Tworzy strukturę MQXQH zawierającą osadzony deskryptor komunikatu
- Dodaje MQMDE, jeśli jest potrzebny i nie jest jeszcze obecny.
- Dołącza dane komunikatu aplikacji
- Umieszcza komunikat w odpowiedniej kolejce transmisji

Umieszczanie komunikatów bezpośrednio w kolejkach transmisji

Aplikacja może również umieścić komunikat bezpośrednio w kolejce transmisji. W takim przypadku aplikacja musi poprzedzić dane komunikatu aplikacji ze strukturą MQXQH i zainicjalizować pola odpowiednimi wartościami. Oprócz tego pole *MDFMT* w parametrze *MSGDSC* wywołania MQPUT lub MQPUT1 musi mieć wartość FMXQH.

Dane znakowe w strukturze MQXQH utworzonej przez aplikację muszą znajdować się w zestawie znaków lokalnego menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), a dane całkowite muszą znajdować się w rodzimym kodowaniu komputera. Ponadto dane znakowe w strukturze MQXQH muszą być dopełniane spacjami do zdefiniowanej długości pola; dane nie mogą być kończone przedwcześnie za pomocą znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca wartości NULL i kolejnych znaków w puste miejsca w strukturze MQXQH.

Należy jednak pamiętać, że menedżer kolejek nie sprawdza, czy struktura MQXQH jest obecna lub czy określono poprawne wartości dla pól.

Pobieranie komunikatów z kolejek transmisji

Aplikacje, które uzyskują komunikaty z kolejki transmisji, muszą przetwarzać informacje w strukturze MQXQH w odpowiedni sposób. Obecność struktury MQXQH na początku danych komunikatu aplikacji jest wskazywana przez wartość FMXQH zwracaną w polu *MDFMT* w parametrze *MSGDSC* wywołania MQGET. Wartości zwracane w polach *MDCSI* i *MDENC* w parametrze *MSGDSC* wskazują zestaw znaków i kodowanie danych znakowych i całkowitoliczbowych w strukturze MQXQH. Zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane za pomocą pól *MDCSI* i *MDENC* w osadzonym deskrypcorze komunikatu.

Pola

Struktura MQXQH zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

XQMD (MQMD1)

Oryginalny deskryptor komunikatu.

Jest to osadzony deskryptor komunikatu i jest to bliska kopia deskryptora komunikatu MQMD, która została określona jako parametr *MSGDSC* w wywołaniu MQPUT lub MQPUT1, gdy komunikat został pierwotnie umieszczony w kolejce zdalnej.

Uwaga: Jest to deskryptor MQMD w wersji version-1.

Wartości początkowe pól w tej strukturze są takie same, jak wartości w strukturze MQMD.

XQRQ (48-bajtowy łańcuch znaków)

Nazwa kolejki docelowej.

Jest to nazwa kolejki komunikatów, która jest pozornym miejscem docelowym dla komunikatu (może to okazać się, że nie jest to rzeczywiste docelowe miejsce docelowe, jeśli na przykład kolejka ta jest zdefiniowana w produkcie *XQRQM* jako lokalna definicja innej kolejki zdalnej).

Jeśli komunikat jest komunikatem listy dystrybucyjnej (to znaczy polem *MDFMT* w deskrypcorze osadzonego komunikatu jest FMDH), pole *XQRQ* jest puste.

Długość tego pola jest podana przez LNQN. Początkowa wartość tego pola to 48 znaków odstępu.

XQRQM (48-bajtowy łańcuch znaków)

Nazwa docelowego menedżera kolejek.

Jest to nazwa menedżera kolejek lub grupy współużytkowania kolejki, która jest właścicielem kolejki, która jest jawnym miejscem docelowym dla komunikatu.

Jeśli komunikat jest komunikatem z listą dystrybucyjną, pole *XQRQM* jest puste.

Długość tego pola jest podana przez *LNQMN*. Początkowa wartość tego pola to 48 znaków odstępu.

XQSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

XQSIDV

Identyfikator struktury nagłówka kolejki transmisji.

Wartością początkową tego pola jest XQSIDV.

XQVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

XQVER1

Numer wersji struktury nagłówka kolejki transmisji.

Następująca stała określa numer wersji bieżącej wersji:

XQVERC

Bieżąca wersja struktury nagłówka kolejki transmisji.

Początkowa wartość tego pola to XQVER1.

Wartości początkowe

Tabela 168. Początkowe wartości pól w MQXQH		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>XQSID</i>	XQSIDV	'XQH¬'
<i>XQVER</i>	XQVER1	1
<i>XQRQ</i>	Brak	Puste
<i>XQRQM</i>	Brak	Puste
<i>XQMD</i>	Te same nazwy i te same wartości co MQMD; patrz Tabela 148 na stronie 1168	-
Uwagi:		
1. Symbol ¬ reprezentuje pojedynczy pusty znak.		

Deklaracja RPG

```
D*.1.....2.....3.....4.....5.....6.....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID          1      4    INZ('XQH ')
D* Structure version number
D  XQVER          5      8I 0 INZ(1)
D* Name of destination queue
D  XQRQ           9      56    INZ
D* Name of destination queue manager
D  XQRQM          57     104    INZ
D* Original message descriptor
```

D	XQ1SID	105	108	INZ('MD')
D	XQ1VER	109	112I 0	INZ(1)
D	XQ1REP	113	116I 0	INZ(0)
D	XQ1MT	117	120I 0	INZ(8)
D	XQ1EXP	121	124I 0	INZ(-1)
D	XQ1FB	125	128I 0	INZ(0)
D	XQ1ENC	129	132I 0	INZ(273)
D	XQ1CSI	133	136I 0	INZ(0)
D	XQ1FMT	137	144	INZ('')
D	XQ1PRI	145	148I 0	INZ(-1)
D	XQ1PER	149	152I 0	INZ(2)
D	XQ1MID	153	176	INZ(X'0000000000000000-0000000000000000000000-000000000000')
D				
D	XQ1CID	177	200	INZ(X'0000000000000000-0000000000000000000000-000000000000')
D				
D	XQ1BOC	201	204I 0	INZ(0)
D	XQ1RQ	205	252	INZ
D	XQ1RM	253	300	INZ
D	XQ1UID	301	312	INZ
D	XQ1ACC	313	344	INZ(X'0000000000000000-0000000000000000000000-0000000000000000000000-0000000')
D				
D	XQ1AID	345	376	INZ
D	XQ1PAT	377	380I 0	INZ(0)
D	XQ1PAN	381	408	INZ
D	XQ1PD	409	416	INZ
D	XQ1PT	417	424	INZ
D	XQ1AOD	425	428	INZ

Wywołania funkcji

Informacje zawarte w tej sekcji umożliwiają zapoznanie się z wywołaniami funkcji dostępnymi w programowaniu produktu IBM i .

Konwencje używane w opisach wywołań w systemie IBM i

W przypadku każdego wywołania ta kolekcja tematów zawiera opis parametrów i sposobu użycia wywołania. Następuje to po typowych wywołaniach wywołania, oraz typowych deklaracjach jego parametrów, w języku programowania RPG.

Ważne: Podczas kodowania wywołań interfejsu API produktu IBM MQ należy upewnić się, że zostały podane wszystkie odpowiednie parametry (opisane w poniższych sekcjach). Niewykonalne działanie może spowodować nieprzewidywalne rezultaty.

Opis każdego wywołania zawiera następujące sekcje:

Nazwa połączenia

Nazwa połączenia, po której następuje krótki opis celu wywołania.

Parametry

W przypadku każdego parametru po nazwie występuje jego typ danych w nawiasach () i jego kierunek; na przykład:

CMPCOD (9-cyfrowa liczba całkowita)-dane wyjściowe

Istnieje więcej informacji na temat typów danych struktury w produkcie [“Elementarne typy danych”](#) na stronie 1010.

Kierunek parametru może być następujący:

Wejście

Ten parametr musi być podany przez użytkownika (programista).

Wyjście

Wywołanie zwraca ten parametr.

Wejście/wyjście

Ten parametr należy podać, ale jest on modyfikowany przez wywołanie.

Dostępny jest również krótki opis przeznaczenia parametru wraz z listą dowolnych wartości, jakie może przyjmować parametr.

Ostatnie dwa parametry w każdym wywołaniu to kod zakończenia i kod przyczyny. Kod zakończenia wskazuje, czy wywołanie zostało zakończone pomyślnie, częściowo, czy nie. Dodatkowe informacje na temat częściowego powodzenia lub niepowodzenia wywołania są podane w kodzie przyczyny.

Użycie notatek

Dodatkowe informacje na temat połączenia, opisujące, jak go używać oraz wszelkie ograniczenia w jego stosowaniu.

Wywołanie RPG

Typowe wywołanie wywołania oraz deklaracja jego parametrów, w języku RPG.

Inne konwencje notacyjne to:

State

Nazwy stałych są wyświetlane wielkimi literami, na przykład OOOOUT.

Tablice

W niektórych wywołaniach parametry są tablicami łańcuchów znaków o rozmiarze, który nie jest stały. W opisach tych parametrów małe litery *n* oznaczają stałą numeryczną. Podczas kodowania deklaracji dla tego parametru należy zastąpić wartość *n* wartością liczbową, która jest wymagana.

MQBACK-wycofanie zmian

Wywołanie MQBACK wskazuje menedżerowi kolejek, że wszystkie operacje pobierania i umieszczania komunikatów wystąpiły od ostatniego punktu synchronizacji, dla którego ma zostać wykonana kopia zapasowa. Komunikaty umieszczone jako część jednostki pracy są usuwane; komunikaty pobrane jako część jednostki pracy są ponownie umieszczane w kolejce.

- W systemie IBM iwywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

- [“Składnia” na stronie 1271](#)
- [“Użycie notatek” na stronie 1271](#)
- [“Parametry” na stronie 1273](#)
- [“Deklaracja RPG” na stronie 1274](#)

Składnia

MQBACK (*Hconn*, *CompCode*, *Reason*)

Użycie notatek

Te uwagi dotyczące użycia można rozważyć podczas używania komendy MQBACK.

1. To wywołanie może być używane tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Jest to lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ.
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, zamiast MQBACK należy użyć odpowiedniego wywołania z powrotem. Środowisko może również obsługiwać niejawne wycofania spowodowane przez działanie aplikacji nieprawidłowo kończące się.
 - W systemie IBM iwywołanie to może być używane dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE (*JOB) nie może zostać wydana dla zadania.

3. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek”](#) na stronie 1312.

4. Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwytami kolejki i obejmują takie elementy jak:

- Wartości pól *MDGID*, *MDSEQ*, *MDOFF* i *MDMFL* w strukturze MQMD.
- Określa, czy komunikat jest częścią jednostki pracy.
- W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje *trzy* zestawy informacji o grupach i segmentach, jeden zestaw dla każdego z następujących elementów:

- Ostatnie pomyślne wywołanie MQPUT (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które przeglądało komunikat w kolejce (ten *nie może* być częścią jednostki pracy).

Jeśli aplikacja wstawi lub pobiera komunikaty jako część jednostki pracy, a następnie aplikacja podejmie decyzję o wytworzeniu kopii zapasowej jednostki pracy, informacje o grupie i segmencie zostaną odtworzone do wartości, którą poprzednio:

- Informacje powiązane z wywołaniem MQPUT są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
- Informacje powiązane z wywołaniem MQGET są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zakresem jednostki pracy, nie mają informacji o grupach i segmentach, które zostały odtworzone w przypadku, gdy tworzona jest kopia zapasowa jednostki pracy.

Odtwarzanie informacji o grupach i segmentach do jej poprzedniej wartości, gdy tworzona jest kopia zapasowa jednostki pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy, a także zrestartowanie w poprawnym punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczające informacje, aby móc restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu. Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu można znaleźć w sekcji PMLOGO opisanej w sekcji [“MQPMO-Put-message, opcje”](#) na stronie 1191 oraz w opcji GMLOGO opisanej w sekcji [“MQGMO-Opcje Get-message”](#) na stronie 1089.

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

1. Jednostka pracy ma ten sam zasięg co uchwyt połączenia. Oznacza to, że wszystkie wywołania produktu IBM MQ, które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Więcej informacji na temat zasięgu uchwytów połączeń zawiera opis parametru *HCONN* opisanego w sekcji [“MQCONN-Połączenie menedżera kolejek”](#) na stronie 1298.
2. To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.

3. Długotrwa aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może spowodować zapętnienie kolejek komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć tę możliwość, administrator powinien ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapętnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duże, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQBACK ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *COMCOD*.

Jeśli *COMCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *COMCOD* to CCFail:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2123

(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQBACK(HCONN : COMCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
D*MQBACK                                PR      EXTPROC('MQBACK')  
D* Connection handle  
D HCONN                                10I 0 VALUE  
D* Completion code  
D COMCOD                                10I 0  
D* Reason code qualifying COMCOD  
D REASON                                10I 0
```

IBM i

AIX, HP-UX, IBM i, Solaris, Windows S/390

MQBEGIN-Rozpoczęcie jednostki pracy

Wywołanie MQBEGIN rozpoczyna jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zewnętrzne menedżery zasobów.

- To wywołanie jest obsługiwane w następujących środowiskach: AIX, HP-UX, IBM i, Solaris, Windows.
- [“Składnia” na stronie 1274](#)
- [“Użycie notatek” na stronie 1274](#)
- [“Parametry” na stronie 1275](#)
- [“Deklaracja RPG” na stronie 1277](#)

Składnia

MQBEGIN (*HCONN*, *BEGOP*, *CMPCOD*, *REASON*)

Użycie notatek

1. Wywołanie MQBEGIN może zostać użyte do uruchomienia jednostki pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zmiany zasobów należących do innych menedżerów zasobów. Menedżer kolejek obsługuje trzy typy jednostek pracy:

Menedżer kolejek-koordynowana lokalna jednostka pracy

Jest to jednostka pracy, w której menedżer kolejek jest jedynym uczestniczącym menedżerem zasobów, a więc menedżer kolejek działa jako koordynator jednostki pracy.

- Aby uruchomić ten typ jednostki pracy, opcja PMSYP lub GMSYP powinna zostać określona w pierwszej wywołaniu MQPUT, MQPUT1 lub MQGET w jednostce pracy.

Nie jest konieczne, aby aplikacja wywołała wywołanie komendy MQBEGIN w celu uruchomienia jednostki pracy, ale jeśli używana jest opcja MQBEGIN, wywołanie kończy się z CCWARN i kodem przyczyny RC2121.

- Aby można było zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołania MQCMIT lub MQBACK.

Menedżer kolejek-koordynowana globalna jednostka pracy

Jest to jednostka pracy, w której menedżer kolejek działa jako koordynator jednostki pracy, zarówno w przypadku zasobów IBM MQ, *jak i* dla zasobów należących do innych menedżerów zasobów. Te menedżery zasobów współpracują z menedżerem kolejek w celu zapewnienia, że wszystkie zmiany w zasobach w jednostce pracy są zatwierdzane lub wycofane razem.

- Aby uruchomić ten typ jednostki pracy, należy użyć wywołania MQBEGIN.
- Aby można było zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań MQCMIT i MQBACK.

Zewnętrznie koordynowana globalna jednostka pracy

Jest to jednostka pracy, w której menedżer kolejek jest uczestnikiem, ale menedżer kolejek nie działa jako koordynator jednostki pracy. Zamiast tego istnieje zewnętrzny koordynator jednostki pracy, z którym współpracuje menedżer kolejek.

- Aby rozpocząć ten typ jednostki pracy, należy użyć odpowiedniego wywołania udostępnionego przez zewnętrznego koordynatora jednostki pracy.

Jeśli wywołanie MQBEGIN jest używane do podjęcia próby uruchomienia jednostki pracy, wywołanie nie powiedzie się i zostanie użyty kod przyczyny RC2012.

- Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań zatwierdzenia i zaplecza, które są udostępniane przez zewnętrznego koordynatora jednostki pracy.

Jeśli wywołanie MQCMIT lub MQBACK jest używane do próby zatwierdzenia lub wycofania jednostki pracy, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2012.

2. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek”](#) na stronie 1312.
3. Aplikacja może w danym momencie uczestniczyć tylko w jednej jednostce pracy. Wywołanie funkcji MQBEGIN kończy się niepowodzeniem z kodem przyczyny RC2128, jeśli istnieje już jednostka pracy dla aplikacji, niezależnie od tego, jaki typ jednostki pracy jest taki sam.
4. Wywołanie MQBEGIN nie jest poprawne w środowisku klienta IBM MQ. Próba użycia wywołania nie powiodła się. Kod przyczyny: RC2012.
5. Gdy menedżer kolejek działa jako koordynator jednostki pracy dla globalnych jednostek pracy, menedżerowie zasobów, którzy mogą uczestniczyć w jednostce pracy, są zdefiniowani w pliku konfiguracyjnym menedżera kolejek.
6. W systemie IBM następujące trzy typy jednostek pracy są obsługiwane w następujący sposób:
 - **Lokalne jednostki pracy koordynowane przez menedżera kolejek** mogą być używane tylko wtedy, gdy definicja kontroli transakcji nie istnieje na poziomie zadania, tj. komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) nie może zostać wydana dla zadania.
 - **Globalne jednostki pracy koordynowane przez menedżera kolejek** nie są obsługiwane.
 - **Zewnętrznie-koordynowane globalne jednostki pracy** mogą być używane tylko wtedy, gdy definicja kontroli transakcji istnieje na poziomie zadania, tj. komenda STRCMTCTL z parametrem CMTSCOPE(*JOB) musi zostać wydana dla zadania. Jeśli ta operacja została wykonana, operacje IBM i COMMIT i ROLLBACK mają zastosowanie do zasobów produktu IBM MQ, a także do zasobów należących do innych uczestniczących menedżerów zasobów.

Parametry

Wywołanie MQBEGIN ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość HCONN została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

BEGOP (MQBO)-wejście/wyjście

Opcje, które sterują działaniem komendy MQBEGIN.

Szczegółowe informacje na ten temat zawiera sekcja [“MQBO-opcje rozpoczęcia”](#) na stronie 1032.

Jeśli nie są wymagane żadne opcje, programy napisane w języku C lub S/390 assembler mogą określać pusty adres parametru, zamiast określać adres struktury MQBO.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2121

(2121, X'849 ') Nie zarejestrowano żadnych uczestniczących menedżerów zasobów.

RC2122

(2122, X'84A') Uczestniczy menedżer zasobów nie jest dostępny.

Jeśli *CMPCOD* to CCFAIL:

RC2134

(2134, X'856 ') Struktura opcji Begin-options jest niepoprawna.

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2012

(2012, X'7DC') Wywołanie nie jest poprawne w środowisku.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2128

(2128, X'850 ') Jednostka pracy już została uruchomiona.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQBEGIN(HCONN : BEGOP : CMPCOD :  
C                                REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQBEGIN          PR          EXTPROC('MQBEGIN')  
D* Connection handle  
D HCONN           10I 0 VALUE  
D* Options that control the action of MQBEGIN  
D BEGOP           12A  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CMPCOD  
D REASON          10I 0
```

MQBUFMH-przekształcanie buforu w uchwyt komunikatu

Wywołanie funkcji MQBUFMH przekształca bufor w uchwyt komunikatu i jest odwrotnym wywołaniem wywołania MQMHBUF.

To wywołanie pobiera deskryptor komunikatu i właściwości MQRFH2 w buforze i udostępnia je za pomocą uchwytu komunikatu. Właściwości MQRFH2 w danych komunikatu są, opcjonalnie, usuwane. Pola *Encoding, CodedCharSetIdi Format* w deskrytorze komunikatu są aktualizowane, jeśli jest to konieczne, aby poprawnie opisać zawartość buforu po usunięciu właściwości.

- [“Składnia” na stronie 1277](#)
- [“Użycie notatek” na stronie 1277](#)
- [“Parametry” na stronie 1277](#)
- [“Deklaracja RPG” na stronie 1279](#)

Składnia

MQBUFMH (*Hconn, Hmsg, BufMsgHOpts, MsgDesc, Buffer, BufferLength, DataLength, CompCode, Reason*)

Użycie notatek

Wywołania MQBUFMH nie mogą zostać przechwycone przez wyjścia funkcji API-bufor jest przekształcany w uchwyt komunikatu w obszarze aplikacji; wywołanie nie dociera do menedżera kolejek.

Parametry

Wywołanie MQBUFMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* musi być zgodna z uchwyttem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony za pomocą komendy HCUNAS, należy ustanowić poprawne połączenie w wątku przekształcając bufor w uchwyt komunikatu. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie zakończy się niepowodzeniem z kodem RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt jest uchwyttem komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

BMHOPT (MQBMHO)-dane wejściowe

Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki uchwyt komunikatów są generowane z buforów.

Szczegółowe informacje na ten temat zawiera sekcja ["MQBMHO-Opcje uchwytu buforu do obsługi komunikatów"](#) na stronie 1030.

MSGDSC (MQMD)-wejście/wyjście

Struktura MSGDSC zawiera właściwości deskryptora komunikatu i opisuje zawartość obszaru buforu.

W przypadku wyjścia z wywołania właściwości są opcjonalnie usuwane z obszaru buforu, a w tym przypadku deskryptor komunikatu jest aktualizowany w celu poprawnego opisanie obszaru buforu.

Dane w tej strukturze muszą znajdować się w zestawie znaków i kodowaniu aplikacji.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

BUFLEN to długość obszaru buforu (w bajtach).

Wartość BUFLEN równa zero bajtów jest poprawna i wskazuje, że obszar buforu nie zawiera żadnych danych.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście/wyjście

BUFFER definiuje obszar zawierający bufor komunikatów. W przypadku większości danych, należy wyrównać bufor na granicy 4-bajtowej.

Jeśli BUFFER zawiera dane znakowe lub numeryczne, ustaw wartości pól *CodedCharSetId* i *Encoding* w parametrze MSGDSC na wartości odpowiednie dla danych. Umożliwia to przekształcenie danych, jeśli to konieczne.

Jeśli właściwości znajdują się w buforze komunikatów, są one opcjonalnie usuwane, a później stają się dostępne z uchwytu komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że adres dowolnego typu danych może być określony jako parametr.

If the BUFLEN parameter is zero, BUFFER is not referred to. W tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

DATLEN to długość (w bajtach) buforu, który może mieć usunięte właściwości.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący CMPCOD.

Jeśli CMPCOD to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli CMPCOD to CCFail:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2489

(2489, X'09B9') Struktura opcji bufora dla uchwytu komunikatu nie jest poprawna.

RC2004

(2004, X'07D4') Parametr buforu nie jest poprawny.

RC2005

(2005, X'07D5') Parametr długości buforu nie jest poprawny.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2460

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

RC2026

(2026, X'07EA') deskryptor komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2334

(2334, X'091E') Struktura MQRFH2 nie jest poprawna.

RC2421

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```

C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQBUFMH(HCONN : HMSG : BMHOPT :
                                           MSGDSC : BUFLN : BUFFER :
                                           DATLEN : CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

DMQBUFMH          PR                      EXTPROC('MQBUFMH')
D* Connection handle
D HCONN                      10I 0
D* Message handle
D HMSG                      10I 0
D* Options that control the action of MQBUFMH
D BMHOPT                      12A  VALUE
D* Message descriptor
D MSGDSC                      364A
D* Length in bytes of the Buffer area
D BUFLN                      10I 0
D* Area to contain the message buffer
D BUFFER                      *  VALUE
D* Length of the output buffer
D DATLEN                      10I 0
D* Completion code
D CMPCOD                      10I 0
D* Reason code qualifying CompCode
D REASON                      10I 0

```

MQCB-zarządzanie wywołaniem zwrotnym

Wywołanie MQCB przekierowuje wywołanie zwrotne dla podanego uchwytu obiektu i steruje aktywacją i zmianami w wywołaniu zwrotnym.

Wywołanie zwrotne jest to fragment kodu (określony jako nazwa funkcji, która może być dynamicznie dowiązana lub jako wskaźnik funkcji), która jest wywoływana przez składnik IBM MQ po wystąpieniu określonych zdarzeń.

Aby można było używać funkcji MQCB i MQCTL na kliencie V7, należy połączyć się z serwerem V7, a parametr **SHARECNV** kanału musi mieć wartość niezerową.

Więcej informacji na temat globalnych jednostek pracy można znaleźć w sekcji [Globalne jednostki pracy](#).

Typy wywołań zwrotnych, które mogą być zdefiniowane, to:

Konsument komunikatu

Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat, spełniający określone kryteria wyboru, jest dostępny na uchwycie obiektu.

Tylko jedna funkcja zwrotna może być zarejestrowana dla każdego uchwytu obiektu. Jeśli pojedyncza kolejka ma być odczytywana z wieloma kryteriami wyboru, kolejka musi być otwierana wiele razy, a funkcja konsumenta zarejestrowana na każdym uchwycie.

procedura obsługi zdarzeń

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko wywołań zwrotnych.

Funkcja jest wywoływana, gdy wystąpi warunek zdarzenia, na przykład menedżer kolejek lub połączenie zatrzymujące się lub wyciszające.

Funkcja nie jest wywoływana w przypadku warunków, które są specyficzne dla pojedynczego konsumenta komunikatów, na przykład RC2016; , jednak jest wywoływana, jeśli funkcja zwrotna nie zakończy się normalnie.

- ["Składnia" na stronie 1280](#)
- ["Uwagi dotyczące użycia dla bazy MQCB" na stronie 1280](#)
- ["Parametry dla obiektu MQCB" na stronie 1282](#)
- ["Deklaracja RPG" na stronie 1288](#)

Składnia

MQCB (*HCONN*, *OPERATN*, *HOBJ*, *CBDSC*, *MSGDSC*, *GMO*, *CMPCOD*, *REASON*)

Uwagi dotyczące użycia dla bazy MQCB

1. Baza MQCB jest używana do definiowania działania, które ma być wywoływane dla każdego komunikatu, zgodnego z podanymi kryteriami, dostępnymi w kolejce. Po przetworzeniu działania komunikat jest usuwany z kolejki i przekazywany do zdefiniowanego konsumenta komunikatów lub udostępniany jest znacznik komunikatu, który jest używany do pobierania komunikatu.
2. Obiekt MQCB może być używany do definiowania procedur zwrotnych przed rozpoczęciem korzystania z komendy MQCTL lub z poziomu procedury zwrotnej.
3. Aby użyć obiektu MQCB z poziomu poza procedurą wywołania zwrotnego, należy najpierw zawiesić wykorzystanie komunikatów za pomocą komendy MQCTL i wznowić korzystanie z niej.

Sekwencja wywołań zwrotnych konsumenta komunikatów

Istnieje możliwość skonfigurowania konsumenta w taki sposób, aby wywoływało wywołanie zwrotne w kluczowych punktach podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- gdy połączenie jest uruchomione,

- gdy połączenie jest zatrzymane i
- gdy konsument jest wyrejestrowany, jawnie lub niejawnie przez operację MQCLOSE.

Tabela 169. Definicje komend MQCTL	
Czasownik	Znaczenie
MQCTL (POCZĄTEK)	Wywołanie MQCTL przy użyciu operacji CTLSR
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji CTLSP
MQCTL (WAIT)	Wywołanie MQCTL przy użyciu operacji CTLSW

Umożliwia konsumentowi utrzymanie stanu powiązanego z konsumentem. Jeśli aplikacja zażądała wywołania zwrotnego, reguły wywołania konsumenta są następujące:

Zarejestruj

Jest zawsze pierwszym typem wywołania zwrotnego.

Jest zawsze wywoływany w tym samym wątku, co wywołanie MQCB (CBREG).

START

Jest zawsze wywoływana synchronicznie za pomocą komendy MQCTL (START).

- Wszystkie wywołania zwrotne START są zakończone przed zwrotami komendy MQCTL (START).

Znajduje się w tym samym wątku, co dostarczanie komunikatów, jeśli zażądano CTLTHR.

Wywołanie z uruchomieniem nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne MQCTL (STOP) podczas operacji MQCTL (START) zostanie uruchomione.

STOP

Po wywołaniu tego połączenia nie zostaną dostarczone żadne dodatkowe komunikaty ani żadne zdarzenia, dopóki połączenie nie zostanie zrestartowane.

Wartość STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana na potrzeby START, lub komunikat lub zdarzenie.

DEREGISTER

Jest zawsze ostatnim typem wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie oparte na wątkach w wywołaniach zwrotnych START i STOP. Inicjowanie i czyszczenie oparte na wątkach można wykonywać przy użyciu wywołań zwrotnych REGISTER i DEREGISTER.

Nie należy podejmować żadnych założeń dotyczących życia i dostępności wątku innego niż to, co jest określone. Na przykład, nie należy polegać na wątku pozostającym przy życiu poza ostatnim wywołaniem DEREGISTER. Podobnie, jeśli nie wybrano opcji CTLTHR, nie należy zakładać, że wątek istnieje za każdym razem, gdy połączenie jest nawiązane.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątku, może zawsze utworzyć wątek, a następnie użyć komendy MQCTL (WAIT). W tym kroku *darowizny* wątek do IBM MQ służy do asynchronicznego dostarczania komunikatów.

Użycie połączenia konsumenta komunikatów

Zwykle, gdy aplikacja wysyła kolejne wywołanie MQI, a jedno jest zaległe, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2219.

Istnieją jednak specjalne przypadki, w których aplikacja musi wydać kolejne wywołanie MQI przed zakończeniem poprzedniego wywołania. Na przykład, konsument może być wywoływany podczas wywołania MQCB z CBRE.

W takiej instancji, gdy w wyniku aplikacji wywołujących komendę MQCB lub MQCTL aplikacja zostanie wywołana z powrotem, aplikacja może wydać dalsze wywołanie MQI. Ta instancja oznacza, że można wprowadzić, na przykład, wywołanie MQOPEN, w funkcji konsumenta, gdy jest wywoływana z typem CBCCALLT CBCTRC. Dozwolone jest dowolne wywołanie MQI, z wyjątkiem wywołania MQDISC.

Parametry dla obiektu MQCB

Wywołanie MQCB ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Zarządzanie funkcją wywołania zwrotnego-parametr HCONN.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a ponadto można określić następującą wartość specjalną dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

OPERATN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Zarządzanie funkcją wywołania zwrotnego-parametr OPERATN.

Operacja jest przetwarzana dla wywołania zwrotnego zdefiniowanego dla podanego uchwytu obiektu. Należy określić jedną z następujących opcji; jeśli wymagana jest więcej niż jedna opcja, wartości można dodawać (nie dodawać tej samej stałej więcej niż raz) lub łączyć je za pomocą operacji bitowych OR (jeśli język programowania obsługuje operacje bitowe).

Podane kombinacje nie są poprawne; wszystkie pozostałe kombinacje są poprawne.

CBREG

Zdefiniuj funkcję zwrotną dla podanego uchwytu obiektu. Ta operacja definiuje funkcję, która ma zostać wywołana, oraz kryteria wyboru, które mają być używane.

Jeśli funkcja zwrotna jest już zdefiniowana dla uchwytu obiektu, definicja jest zastępowana. Jeśli podczas zastępowania wywołania zwrotnego zostanie wykryty błąd, funkcja jest wyrejestrowana.

Jeśli wywołanie zwrotne jest zarejestrowane w tej samej funkcji zwrotnej, w której została wcześniej wyrejestrowana, jest ona traktowana jako operacja zastępowania. W przypadku wywołań początkowych lub końcowych nie są wywoływane.

Można użyć CBREG z CTLSU lub CTLRE.

CBUNR

Zatrzymaj konsumowanie komunikatów dla uchwytu obiektu i usuwa uchwyt z tych zakwalifikowanych do wywołania zwrotnego.

Wywołanie zwrotne jest automatycznie wyrejestrowywane, jeśli powiązany uchwyt jest zamknięty.

Jeśli funkcja CBUNR jest wywoływana z poziomu konsumenta, a wywołanie zwrotne ma zdefiniowane wywołanie zatrzymania, jest ono wywoływane po powrocie z konsumenta.

Jeśli ta operacja zostanie wywołana dla *Hobj* bez zarejestrowanego konsumenta, wywołanie zostanie zwrócone z RC2448.

CTLSU

Zawiesza korzystanie z komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie otrzymała zdarzeń podczas zawieszania, a wszystkie zdarzenia, które nie zostały pominięte w stanie zawieszenia, nie zostaną udostępnione operacji po jej wznowieniu.

Po zawieszeniu funkcja konsumenta kontynuuje wywoływanie zwrotnych typów sterowania.

CTLRE

Wznów korzystanie z komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie otrzymała zdarzeń podczas zawieszania, a wszystkie zdarzenia, które nie zostały pominięte w stanie zawieszenia, nie zostaną udostępnione operacji po jej wznowieniu.

CBDSC (MQCBD)-dane wejściowe

Zarządzanie funkcją zwrotną-parametr CBDSC.

Jest to struktura identyfikująca funkcję zwrotną, która jest rejestrowana przez aplikację, oraz opcje używane podczas rejestrowania.

Szczegółowe informacje na temat struktury zawiera sekcja [“MQCBD-deskryptor wywołania zwrotnego”](#) na stronie 291.

Deskryptor wywołania zwrotnego jest wymagany tylko w przypadku opcji CBREG. Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Zarządzanie funkcją wywołania zwrotnego-parametr HOBJ.

Ten uchwyt reprezentuje dostęp, który został utworzony dla obiektu, z którego komunikat ma zostać zużyty. Jest to uchwyt, który został zwrócony z poprzednich wywołań [MQOPEN](#) lub [MQSUB](#) (w parametrze *HOBJ*).

Produkt *HOBJ* nie jest wymagany podczas definiowania procedury obsługi zdarzeń (CBTEH) i musi być określony jako HONONE.

Jeśli ten *Hobj* został zwrócony z wywołania [MQOPEN](#), kolejka musi zostać otwarta z jedną lub więcej spośród następujących opcji:

- OWINPS
- OWINPX
- POINPQ
- OOBWRW

MSGDSC (MQMD)-dane wejściowe

Zarządzanie funkcją zwrotną-parametr MSGDSC.

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu.

Parametr *MsgDesc* definiuje atrybuty komunikatów wymaganych przez konsumenta, a wersja deskryptora MQMD, która ma zostać przekazana do konsumenta komunikatów.

Opcje *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber* i *Offset* w strukturze MQMD są używane do wyboru komunikatów, w zależności od opcji określonych w parametrze *GetMsgOpts*.

Opcje *Encoding* i *CodedCharSetId* są używane do konwersji komunikatów, jeśli zostanie określona opcja GMCONV.

Szczegółowe informacje na ten temat zawiera sekcja [MQMD](#).

MsgDesc jest używany tylko dla CBREG i, jeśli wymagane są wartości inne niż domyślne dla wszystkich pól. Program *MsgDesc* nie jest używany dla procedury obsługi zdarzeń.

Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Należy zauważyć, że jeśli wielu konsumentów jest zarejestrowanych w tej samej kolejce z nakładającymi się selektorami, wybrany konsument dla każdego komunikatu jest niezdefiniowany.

GMO (MQGMO)-wejście

Zarządzanie funkcją wywołania zwrotnego-parametr GMO.

Opcje sterujące sposobem pobierania komunikatów przez konsument komunikatów.

Wszystkie opcje mają znaczenie zgodnie z opisem w [“MQGMO-Opcje Get-message”](#) na stronie 1089, jeśli są używane w wywołaniu MQGET, z wyjątkiem:

GMSSIG

Ta opcja nie jest dozwolona.

GMBRWF, GMBRWN, GMMBH, GMMBC

Kolejność komunikatów dostarczanych do konsumenta przeglądania jest podyktowana kombinacjami tych opcji. Istotne kombinacje to:

GMBRWF

Pierwszy komunikat w kolejce jest dostarczany wielokrotnie do konsumenta. Jest to przydatne w sytuacji, gdy konsument destrukcyjnie konsumuje komunikat w wywołaniu zwrotnym. Tej opcji należy używać z ostrożnością.

GMBRWN

Konsument otrzymuje każdy komunikat w kolejce, od bieżącej pozycji kursora do momentu osiągnięcia końca kolejki.

GMBRWF + GMBRWN

Kursor zostanie zresetowany do początku kolejki. Następnie konsumentowi podaje się każdy komunikat do momentu, gdy kursor osiągnie koniec kolejki.

GMBRWF + GMMBH lub GMMBC

Rozpoczynając od początku kolejki, konsument otrzymuje pierwszą niezaznaczoną wiadomość w kolejce, która jest następnie oznaczona dla tego konsumenta. Ta kombinacja zapewnia, że konsument może odbierać nowe komunikaty dodane za bieżącym punktem kursora.

GMBRWN + GMMBH lub GMMBC

Począwszy od pozycji kursora, konsument otrzymuje następną niezaznaczoną wiadomość w kolejce, która jest następnie oznaczona dla tego konsumenta. Tej kombinacji należy używać z ostrożnością, ponieważ komunikaty mogą być dodawane do kolejki za bieżącą pozycją kursora.

GMBRWF + GMBRWN + GMMBH lub GMMBC

Ta kombinacja nie jest dozwolona, jeśli jest używana, funkcja zwraca wartość RC2046.

GMNWT, GMWT i GMWI

Te opcje sterują sposobem wywoływania konsumenta.

GMNWT

Konsument nigdy nie jest wywoływany z kodem RC2033. Konsument jest wywoływany tylko w przypadku komunikatów i zdarzeń.

GMWT z zerowym GMWI

Kod RC2033 jest przesyłany do konsumenta tylko wtedy, gdy nie ma żadnych komunikatów i

- konsument został uruchomiony
- Konsument został dostarczony co najmniej jeden komunikat od ostatniego kodu przyczyny braku komunikatów.

Uniemożliwia to konsumentowi odpytywanie w pętli zajętości, gdy określony jest przedział czasu oczekiwania na zero.

GMWT i dodatnia GMWI

Użytkownik jest wywoływany po upływie określonego przedziału czasu oczekiwania z kodem przyczyny RC2033. Wywołanie to jest wykonywane niezależnie od tego, czy do konsumenta dostarczono jakiegokolwiek komunikaty. Pozwala to użytkownikowi na przetwarzanie pulsu lub przetwarzania typu wsadowego.

GMWT i GMWI WIULIM

Określa to nieskończone oczekiwanie przed zwróceniem wartości RC2033. Konsument nigdy nie jest wywoływany z kodem RC2033.

GMO jest używany tylko dla CBREG i, jeśli wymagane są wartości inne niż domyślne dla wszystkich pól. Program *GMO* nie jest używany dla procedury obsługi zdarzeń.

Jeśli opcje nie są wymagane, przekazany adres parametru może mieć wartość NULL.

Jeśli uchwyt właściwości komunikatu jest udostępniany w strukturze MQGMO, kopia jest udostępniana w strukturze MQGMO, która jest przekazywana do wywołania zwrotnego konsumenta. Po powrocie z wywołania MQCB aplikacja może usunąć uchwyt właściwości komunikatu.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Zarządzanie funkcją wywołania zwrotnego-parametr CMPCOD.

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Zarządzanie funkcją wywołania zwrotnego-parametr REASON.

Następujące kody przyczyny są kodami, które menedżer kolejek może zwrócić dla parametru *REASON*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to CCFAIL:

RC2204

(2204, X'89C') Adapter nie jest dostępny.

RC2133

(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2374

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

RC2183

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2005

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

RC2219

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2487

(2487, X'9B7') Niepoprawne pole typu wywołania zwrotnego.

RC2448

(2448, X' 990 ') Nie można wyrejestrować, zawiesić ani wznowić, ponieważ nie ma zarejestrowanej procedury zwrotnej.

RC2486

(2486, X'9B6') Należy podać wartość *CallbackFunction* lub *CallbackName*, ale nie obie jednocześnie.

RC2483

(2483, X'9B3') Niepoprawne pole typu wywołania zwrotnego.

RC2484

(2484, X'9B4') Niepoprawne pole opcji MQCBD.

RC2140

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2217

(2217, X'8A9') Brak uprawnień do połączenia.

RC2202

(2202, X'89A') Połączenie wygaszające.

RC2203

(2203, X'89B') Połączenie jest zamykane.

RC2207

(2207, X'89F') Błąd korelacji identyfikatora.

RC2010

(2010, X'7DA') Parametr długości danych nie jest poprawny.

RC2016

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

RC2351

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

RC2186

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

RC2353

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2259

(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2352

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

RC2247

(2247, X'8C7') Opcje zgodności nie są poprawne.

RC2485

(2485, X'9B4') Niepoprawne pole *MaxMsgLength* .

RC2026

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

RC2497

(2497, X'9C1') Określony punkt wejścia funkcji nie został znaleziony w module.

RC2496

(2496, X'9C0') Znaleziono moduł, jednak jest to niepoprawny typ; nie jest to 32-bitowa, 64-bitowa lub poprawna biblioteka dołączana dynamicznie.

RC2495

(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie został autoryzowany do załadowania.

RC2250

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

RC2331

(2331, X'91B') Użycie znacznika komunikatu nie jest poprawne.

RC2033

(2033, X'7F1') Brak dostępnego komunikatu.

RC2034

(2034, X'7F2') Przeglądaj kursor nie umieszczony na komunikacie.

RC2036

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2206

(2206, X'89E') Niepoprawny kod operacji w wywołaniu API Call.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2193

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2394

(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2069

(2069, X'815 ') Signal outstanding for this handle.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2109

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

RC2024

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

RC2072

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2354

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

RC2355

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

RC2255

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

RC2090

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

RC2256

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

RC2257

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

RC2298

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCB(HCONN : OPERATN : CBDSC :
                                           HOBJ : MSGDSC : GMO :
                                           DATLEN : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
DMQCB          PR          EXTPROC('MQCB')
D* Connection handle
D HCONN          10I 0 VALUE
D* Operation
D OPERATN        10I 0 VALUE
D* Callback descriptor
D CBDSC          180A
D* Object handle
D HOBJ          10I 0 VALUE
D* Message Descriptor
D MSGDSC        364A
D* Get options
D GMO          112A
D* Completion code
D CMPCOD        10I 0
* Reason code qualifying CompCode
D REASON        10I 0
```

MQCLOSE-zamknięcie obiektu

Wywołanie MQCLOSE zrzekało się dostępu do obiektu i jest odwrotnym wywołaniem wywołania MQOPEN.

- [“Składnia” na stronie 1288](#)
- [“Użycie notatek” na stronie 1288](#)
- [“Parametry” na stronie 1290](#)
- [“Deklaracja RPG” na stronie 1295](#)

Składnia

MQCLOSE (*HCONN*, *HOBJ*, *OPTS*, *CMPCOD*, *REASON*)

Użycie notatek

1. Gdy aplikacja wysyła wywołanie MQDISC lub kończy się normalnie lub nieprawidłowo, wszystkie obiekty, które zostały otwarte przez aplikację i są nadal otwarte, są automatycznie zamykane z opcją CONONE.
2. Jeśli zamykany obiekt jest *kolejką*, mają zastosowanie następujące punkty:

- Jeśli operacje w kolejce wykonywane są jako część jednostki pracy, kolejka może zostać zamknięta przed lub po wystąpieniu punktu synchronizacji bez wpływu na wynik punktu synchronizacji.
 - Jeśli kolejka została otwarta za pomocą opcji OOB_{RW}, kursor przeglądania zostanie zniszczony. Jeśli kolejka jest później ponownie otwierana za pomocą opcji OOB_{RW}, tworzony jest nowy kursor przeglądania (patrz opcja OOB_{RW} opisana w MQOPEN).
 - Jeśli komunikat jest obecnie zablokowany dla tego uchwytu w czasie wywołania MQCLOSE, blokada jest zwalniana (patrz opcja GMLK opisana w sekcji “MQGMO-Opcje Get-message” na stronie 1089).
3. Jeśli zamykany obiekt jest *kolejką dynamiczną* (trwałą lub tymczasową), mają zastosowanie następujące punkty:
- W przypadku kolejki dynamicznej można określić opcje CODEL lub COPURG, niezależnie od opcji określonych w odpowiednim wywołaniu MQOPEN.
 - Po usunięciu kolejki dynamicznej wszystkie wywołania MQGET z opcją GMWT, które są zależne z kolejką, są anulowane, a kod przyczyny RC2052 jest zwracany. Zapoznaj się z opcją GMWT opisaną w sekcji “MQGMO-Opcje Get-message” na stronie 1089.

Po usunięciu kolejki dynamicznej wszystkie wywołania (inne niż MQCLOSE), które próbują odwołać się do kolejki przy użyciu wcześniej uzyskanego uchwytu *HOB_J*, nie powiodą się z kodem przyczyny RC2052.

Należy pamiętać, że mimo że nie można uzyskać dostępu do usuniętej kolejki przez aplikacje, kolejka nie jest usuwana z systemu, a powiązane zasoby nie są zwalniane, dopóki nie zostaną zamknięte wszystkie uchwytów odwołujący się do tej kolejki, a wszystkie jednostki pracy, które mają wpływ na kolejkę, zostały zatwierdzone lub wycofane.

- W przypadku usunięcia trwałej kolejki dynamicznej, jeśli uchwyt *HOB_J* określony w wywołaniu MQCLOSE *nie* jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, zostanie wykonane sprawdzenie, czy identyfikator użytkownika, który był używany do sprawdzania poprawności wywołania MQOPEN, jest uprawniony do usunięcia kolejki. Jeśli w wywołaniu MQOPEN została określona opcja OOALTU, to sprawdzany identyfikator użytkownika to *ODA_U*.

To sprawdzenie nie jest wykonywane, jeśli:

- Podany uchwyt jest zwracany przez wywołanie MQOPEN, które utworzyło kolejkę.
- Usuwana kolejka jest tymczasową kolejką dynamiczną.
- Jeśli tymczasowa kolejka dynamiczna jest zamknięta, to jeśli uchwyt *HOB_J* określony w wywołaniu MQCLOSE jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, kolejka jest usuwana. Taka sytuacja występuje niezależnie od opcji zamknięcia określonych w wywołaniu MQCLOSE. Jeśli w kolejce znajdują się komunikaty, są one usuwane. Nie są generowane żadne komunikaty raportów.

Jeśli istnieją niezatwierdzone jednostki pracy, które mają wpływ na kolejkę, kolejka i jej komunikaty są nadal usuwane, ale nie powoduje to, że jednostki pracy nie powiodą się. Jednak, jak opisano wcześniej, zasoby powiązane z jednostkami pracy nie są zwalniane do czasu, aż każda z jednostek pracy nie zostanie zatwierdzona lub wycofana.

4. Jeśli zamykany obiekt jest *listą dystrybucyjną*, mają zastosowanie następujące punkty:

- Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest CONONE; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2046 lub RC2045, jeśli określono jakiekolwiek inne opcje.
- Po zamknięciu listy dystrybucyjnej dla kolejek na liście nie są zwracane pojedyncze kody zakończenia i kody przyczyny-tylko parametry *CMPCOD* i *REASON* wywołania są dostępne dla celów diagnostycznych.

Jeśli wystąpi awaria podczas zamykania jednej z kolejek, menedżer kolejek kontynuuje przetwarzanie i podejmuje próbę zamknięcia pozostałych kolejek na liście dystrybucyjnej. Parametry *CMPCOD* i *REASON* w wywołaniu są następnie ustawiane w celu zwrócenia informacji opisujących niepowodzenie. W związku z tym możliwe jest, że kod zakończenia będzie miał wartość CCFAIL, nawet jeśli większość kolejek została pomyślnie zamknięta. Kolejka, w której wystąpił błąd, nie została zidentyfikowana.

Jeśli wystąpi awaria w więcej niż jednej kolejce, nie jest zdefiniowana, która awaria jest raportowana w parametrach *CMPCOD* i *REASON*.

5. W systemie IBM i, jeśli aplikacja była połączona niejawnie po wydaniu pierwszego wywołania MQOPEN, podczas wykonywania ostatniej operacji MQCLOSE występuje niejawna zmaterializowana tabela MQDISC.

Tylko aplikacje działające w trybie zgodności mogą być połączone niejawnie; inne aplikacje muszą wywołać wywołanie MQCONN lub MQCONNX w celu jawnego nawiązania połączenia z menedżerem kolejek.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Parametry

Wywołanie MQCLOSE ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-input/output

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt, który jest zamykany. Obiekt może być dowolnego typu. Wartość *HOBJ* została zwrócona przez poprzednie wywołanie MQOPEN.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia ten parametr na wartość, która nie jest poprawnym uchwytem dla środowiska. Ta wartość jest następująca:

HOUNUH

Uchwyt obiektu nie do użycia.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje sterujące działaniem komendy MQCLOSE.

Parametr *OPTS* określa sposób zamykania obiektu. Tylko trwałe kolejki dynamiczne i subskrypcje mogą być zamykane w więcej niż jeden sposób. Trwałe kolejki dynamiczne mogą być zachowywane lub usuwane. Są to kolejki z atrybutem *DefinitionType* o wartości QDPERM (patrz atrybut *DefinitionType* opisany w sekcji ["Atrybuty dla kolejek" na stronie 1395](#)). Opcje zamknięcia są podsumowane w tabeli w dalszej części tego tematu.

Trwałe subskrypcje mogą być przechowywane lub usuwane. Te subskrypcje są tworzone przy użyciu wywołania MQSUB z opcją SODUR.

Podczas zamykania uchwytu do zarządzanego miejsca docelowego (jest to parametr *Hobj* zwrócony w wywołaniu MQSUB, który użył opcji SOMAN), menedżer kolejek wyczyści wszystkie niepobrażone publikacje, gdy powiązana subskrypcja również została usunięta. Jest to wykonywane przy użyciu opcji CORMSB w parametrze *Hsub* zwróconej w wywołaniu MQSUB. Należy zauważyć, że CORMSB jest domyślnym zachowaniem w tabeli MQCLOSE dla nietrwałej subskrypcji.

Podczas zamykania uchwytu do niezarządzanego miejsca docelowego, użytkownik jest odpowiedzialny za czyszczenie kolejki, w której wysyłane są publikacje. Zaleca się, aby najpierw zamknąć subskrypcję za pomocą CORMSB, a następnie przetworzyć komunikaty poza kolejką, aż do momentu, w którym nie będzie jeszcze pozostało.

Należy określić jeden (i tylko jeden) z następujących elementów:

Opcje zamykania kolejki dynamicznej

Te opcje sterują sposobem zamykania trwałych kolejek dynamicznych:

CODEL

Usuń kolejkę.

Kolejka jest usuwana, jeśli spełniony jest jeden z poniższych warunków:

- Jest to stała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i brak komunikatów w kolejce i brak niezatwierdzonych żądań pobierania lub umieszczania żądań dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło HOBj. W takim przypadku wszystkie komunikaty znajdujące się w kolejce są usuwane.

We wszystkich innych przypadkach, w tym w przypadku, gdy produkt *Hobj* został zwrócony w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2045, a obiekt nie został usunięty.

COPURG

Usuń kolejkę, wyczyszczając na niej wszystkie komunikaty.

Kolejka jest usuwana, jeśli spełniony jest jeden z poniższych warunków:

- Jest to stała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i nie ma żadnych niezatwierdzonych żądań pobrania lub umieszczenia oczekujących żądań dla kolejki (dla bieżącego zadania lub innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło HOBj.

We wszystkich innych przypadkach, w tym w przypadku, gdy produkt *Hobj* został zwrócony w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2045, a obiekt nie został usunięty.

W następnej tabeli przedstawiono poprawne opcje zamknięcia oraz informacje o tym, czy obiekt jest zachowywany, czy usuwany.

Tabela 170. Poprawne opcje zamknięcia dla obiektów zatrzymanych lub usuniętych			
Typ obiektu lub kolejki	CONONE	CODEL	COPURG
Obiekt inny niż kolejka	Zachowany	Niepoprawne	Niepoprawne
Kolejka predefiniowana	Zachowany	Niepoprawne	Niepoprawne
stała kolejka dynamiczna	Zachowany	Usunięto, jeśli jest puste i nie ma oczekujących aktualizacji	Komunikaty usunięte; kolejka usunięta, jeśli nie ma oczekujących aktualizacji
Tymczasowa kolejka dynamiczna (wywołanie wydane przez twórcę kolejki)	Usunięte	Usunięte	Usunięte

Tabela 170. Poprawne opcje zamknięcia dla obiektów zatrzymanych lub usuniętych (kontynuacja)			
Typ obiektu lub kolejki	CONONE	CODEL	COPURG
Tymczasowa kolejka dynamiczna (wywołanie nie zostało wysłane przez twórcę kolejki)	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna	Zachowany	Niepoprawne	Niepoprawne
Miejsce docelowe zarządzanego subskrypcji	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna (subskrypcja została usunięta)	Komunikaty usunięte; kolejka usunięta	Niepoprawne	Niepoprawne

Opcje zamknięcia subskrypcji

Te opcje kontrolują, czy trwałe subskrypcje są usuwane po zamknięciu uchwytu, oraz czy publikacje nadal oczekujące na odczytanie przez aplikację są czyszczone. Te opcje są poprawne tylko do użycia z uchwytami obiektu zwróconego w parametrze *Hsub* wywołania *MQSUB*.

COKPSB

Uchwyt do subskrypcji jest zamknięty, ale subskrypcja jest przechowywana. Publikacje będą nadal wysyłane do miejsca docelowego określonego w subskrypcji. Ta opcja jest poprawna tylko wtedy, gdy subskrypcja została wykonana z opcją *SODUR*. *COKPSB* jest wartością domyślną, jeśli subskrypcja jest trwała.

CORMSB

Subskrypcja zostanie usunięta, a uchwyt do subskrypcji jest zamknięty.

Parametr *Hobj* wywołania *MQSUB* nie jest unieważniony przez zamknięcie parametru *Hsub* i może być nadal używany dla operacji *MQGET* lub *MQCB* w celu otrzymywania pozostałych publikacji. Jeśli parametr *Hobj* wywołania *MQSUB* jest również zamknięty, to jeśli jest to zarządzane miejsce docelowe, wszystkie niepobrane publikacje zostaną usunięte.

Wartość *CORMSB* jest wartością domyślną, jeśli subskrypcja nie jest trwała.

Te opcje zamknięcia subskrypcji są podsumowane w następujących tabelach:

Aby zamknąć uchwyt subskrypcji trwałej, ale pozostaw subskrypcję w pobliżu, użyj następujących opcji zamknięcia subskrypcji:

Zadanie	Opcja zamknięcia subskrypcji
Przechowuj publikacje na uchwycie <i>MQOPENed</i>	<i>COKPSB</i>
Usuń publikacje na uchwycie <i>MQOPENed</i>	Działanie jest niedozwolone
Przechowuj publikacje na uchwycie z <i>SOMAN</i>	<i>COKPSB</i>
Usuń publikacje na uchwycie z <i>SOMAN</i>	Działanie jest niedozwolone

Aby anulować subskrypcję, zamykając trwały uchwyt subskrypcji i anulować subskrypcję lub zamknąć uchwyt subskrypcji nietrwałej, użyj następujących opcji zamknięcia subskrypcji:

Zadanie	Opcja zamknięcia subskrypcji
Przechowuj publikacje na uchwycie <i>MQOPENed</i>	<i>CORMSB</i>
Usuń publikacje na uchwycie <i>MQOPENed</i>	Działanie jest niedozwolone
Przechowuj publikacje na uchwycie z <i>SOMAN</i>	<i>CORMSB</i>
Usuń publikacje na uchwycie z <i>SOMAN</i>	<i>COPGSB</i>

Opcje odczytu z wyprzedzeniem

Poniższe opcje kontrolują, co dzieje się z nietrwałymi komunikatami, które zostały wysłane do klienta przed żądaniem ich zgłoszenia i nie zostały jeszcze wykorzystane przez aplikację. Komunikaty te są przechowywane w buforze odczytu z wyprzedzeniem klienta oczekującego na żądanie przez aplikację i mogą zostać usunięte lub skonsumowane z kolejki przed zakończeniem operacji MQCLOSE.

COIMM

Obiekt jest zamykany natychmiast, a wszystkie komunikaty, które zostały wysłane do klienta, zanim aplikacja zażądała ich usunięcia i nie są dostępne do wykorzystania przez żadną aplikację. Jest to wartość domyślna.

COQSC

Żądanie zamknięcia obiektu jest wykonywane, ale jeśli wszystkie komunikaty, które zostały wysłane do klienta przed żądaniem ich przez aplikację, nadal znajdują się w buforze odczytu z wyprzedzeniem klienta, wywołanie MQCLOSE zwróci kod ostrzegawczy RC2458, a uchwyt obiektu pozostanie poprawny.

Aplikacja może następnie nadal używać uchwytu obiektu do pobierania komunikatów aż do momentu, gdy nie będzie już dostępny, a następnie ponownie zamknąć obiekt. Żadne komunikaty nie będą wysyłane do klienta przed żądaniem aplikacji, a następnie odczyt z wyprzedzeniem jest teraz wyłączony.

Zaleca się, aby aplikacje używały COQSC zamiast do punktu, w którym nie ma więcej komunikatów w buforze odczytu z wyprzedzeniem klienta, ponieważ może dojść do komunikatu między ostatnim wywołaniem MQGET a następującą tabelą MQCLOSE, która zostanie odrzucona, jeśli używany jest produkt COIMM.

Jeśli komenda MQCLOSE z COQSC jest wydawana z asynchronicznej funkcji zwrotnej, stosowane jest to samo zachowanie odczytu z wyprzedzeniem. Jeśli zwrócony zostanie kod ostrzegawczy RC2458, funkcja zwrotna zostanie wywołana co najmniej raz. Gdy ostatni pozostały komunikat, który został odczytany z wyprzedzeniem, został przekazany do funkcji zwrotnej, to pole CBCFLG jest ustawione na CBCFBE.

Opcja domyślna

Jeśli nie jest wymagana żadna z opisanych wcześniej opcji, można użyć następującej opcji:

CONONE

Opcjonalne przetwarzanie zamknięcia nie jest wymagane.

Ten *musi* być określony dla:

- Obiekty inne niż kolejki
- Kolejki predefiniowane
- Tymczasowe kolejki dynamiczne (ale tylko w tych przypadkach, w których *HOB*J nie jest to uchwyt zwracany przez wywołanie MQOPEN, które utworzyło kolejkę).
- Lista dystrybucyjna

We wszystkich poprzednich przypadkach obiekt jest zachowywany i nie jest usuwany.

Jeśli ta opcja jest określona dla tymczasowej kolejki dynamicznej:

- Kolejka jest usuwana, jeśli została utworzona za pomocą wywołania MQOPEN, które zwróciło *HOB*J; wszystkie komunikaty, które znajdują się w kolejce, są czyszczone.
- We wszystkich innych przypadkach kolejka (i wszystkie komunikaty) są zachowywane.

Jeśli ta opcja jest określona dla trwałej kolejki dynamicznej, kolejka jest zachowywana i nie została usunięta.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2241

(2241, X'8C1') Grupa komunikatów nie została zakończona.

RC2242

(2242, X'8C2') Komunikat logiczny nie został zakończony.

Jeśli *CMPCOD* to CCFAIL:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2035

(2035, X'7F3') Brak uprawnień do dostępu.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2045

(2045, X'7FD') Opcja nie jest poprawna dla typu obiektu.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2055

(2055, X'807 ') Kolejka zawiera jeden lub więcej komunikatów lub niezatwierdzonych żądań umieszczania lub pobierania.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2063

(2063, X'80F') Wystąpił błąd zabezpieczeń.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCLOSE(HCONN : HOBJ : OPTS :  
C                                CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*.1.....2.....3.....4.....5.....6.....7..  
DMQCLOSE      PR      EXTPROC('MQCLOSE')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Object handle  
D HOBJ              10I 0  
D* Options that control the action of MQCLOSE  
D OPTS              10I 0 VALUE  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CMPCOD  
D REASON            10I 0
```

MQCMIT-zatwierdzanie zmian

Wywołanie MQCMIT wskazuje menedżera kolejek, że aplikacja osiągnęła punkt synchronizacji, oraz że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają zostać wykonane na stałe. Komunikaty umieszczone jako część jednostki pracy są udostępniane innym aplikacjom; komunikaty pobierane jako część jednostki pracy są usuwane.

- W systemie IBM iwywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

- [“Składnia” na stronie 1295](#)
- [“Użycie notatek” na stronie 1295](#)
- [“Parametry” na stronie 1296](#)
- [“Deklaracja RPG” na stronie 1297](#)

Składnia

MQCMIT (HCONN, COMCOD, REASON)

Użycie notatek

Podczas korzystania z produktu MQCMIT należy rozważyć zastosowanie tych uwag dotyczących użycia.

1. To wywołanie może być używane tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Jest to lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, zamiast komendy MQCMIT należy użyć odpowiedniego wywołania zatwierdzenia. Środowisko może również obsługiwać niejawne zatwierdzenie spowodowane przez aplikację kończąca się normalnie.

- W systemie IBM i wywołanie to może być używane dla lokalnych jednostek pracy koordynowanych przez menedżera kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE (*JOB) nie może zostać wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek” na stronie 1312](#).
 4. Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwytami kolejki i obejmują takie elementy jak:
 - Wartości pól *MDGID*, *MDSEQ*, *MDOFF* i *MDMFL* w strukturze MQMD.
 - Określa, czy komunikat jest częścią jednostki pracy.
 - W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Gdy jednostka pracy jest zatwierdzana, menedżer kolejek zachowuje informacje o grupie i segmencie, a aplikacja może kontynuować wprowadzanie lub pobieranie komunikatów w bieżącej grupie komunikatów lub w komunikacie logicznym.

Zachowywanie informacji o grupach i segmentach, gdy zatwierdzana jest jednostka pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczające informacje, aby móc restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu. Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu można znaleźć w sekcji PMLOGO opisanej w sekcji [“MQPMO-Put-message, opcje” na stronie 1191](#) oraz w opcji GMLOGO opisanej w sekcji [“MQGMO-Opcje GetMessage” na stronie 1089](#).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

1. Jednostka pracy ma ten sam zasięg co uchwyt połączenia. Oznacza to, że wszystkie wywołania produktu IBM MQ, które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Więcej informacji na temat zasięgu uchwytów połączeń zawiera opis parametru *HCONN* opisanego w tabeli MQCONN.
2. To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.
3. Długotrwa aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub żądania z powrotem, może spowodować zapełnienie kolejek komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć tę możliwość, administrator powinien ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapełnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duża, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQCMIT ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

COMCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *COMCOD*.

Jeśli *COMCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *COMCOD* to CCWARN:

RC2003

(2003, X'7D3') Wytworzona jednostka pracy.

RC2124

(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli *COMCOD* to CCFAIL:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2123

(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCMIT(HCONN : COMCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..
```

DMQCMIT	PR	EXTPROC('MQCMIT')
D* Connection handle		
D HCONN		10I 0 VALUE
D* Completion code		
D COMCOD		10I 0
D* Reason code qualifying COMCOD		
D REASON		10I 0

MQCONN-Połączenie menedżera kolejek

Wywołanie MQCONN łączy program aplikacji z menedżerem kolejek. Udostępnia uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach kolejkowania komunikatów.

- W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Te aplikacje są automatycznie połączone z menedżerem kolejek po wydaniu pierwszego wywołania MQOPEN. Jednak wywołania MQCONN i MQDISC są nadal akceptowane z aplikacji IBM i.

Inne aplikacje (czyli aplikacje, które nie działają w trybie zgodności) muszą używać wywołania MQCONN lub MQCONNX do nawiązywania połączenia z menedżerem kolejek, a wywołanie MQDISC w celu rozłączenia się z menedżerem kolejek. Należy rozważyć użycie tego stylu programowania.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

W systemach IBM MQ for Windows, UNIX i IBM i każdy wątek w aplikacji może łączyć się z różnymi menedżerami kolejek. W innych systemach wszystkie współbieżne połączenia w ramach procesu muszą należeć do tego samego menedżera kolejek.

- “Składnia” na stronie 1298
- “Użycie notatek” na stronie 1298
- “Parametry” na stronie 1299
- “Deklaracja RPG” na stronie 1302

Składnia

MQCONN (QMNAME, HCONN, CMPCOD, REASON)

Użycie notatek

- Menedżer kolejek, do którego nawiąże połączenie przy użyciu wywołania MQCONN, jest nazywany *lokalnym menedżerem kolejek*.
- Kolejki, których właścicielem jest lokalny menedżer kolejek, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie komunikatów z tych kolejek.

Kolejki współużytkowane, których właścicielem jest grupa współużytkowania kolejki, do której należy lokalny menedżer kolejek, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie komunikatów z tych kolejek.

Kolejki, których właścicielem jest zdalne menedżery kolejek, są wyświetlane jako kolejki zdalne. Możliwe jest umieszczanie komunikatów w tych kolejkach, ale nie jest możliwe pobieranie komunikatów z tych kolejek.

- Jeśli menedżer kolejek zakończy się niepowodzeniem podczas działania aplikacji, aplikacja musi ponownie wywołać wywołanie MQCONN, aby uzyskać nowy uchwyt połączenia, który ma być używany podczas kolejnych wywołań produktu IBM MQ. Aplikacja może wywoływać wywołanie MQCONN okresowo, dopóki wywołanie nie powiedzie się.

Jeśli aplikacja nie jest pewna, czy jest połączona z menedżerem kolejek, aplikacja może bezpiecznie wywołać wywołanie MQCONN w celu uzyskania uchwytu połączenia. Jeśli aplikacja jest już połączona, zwracany uchwyt jest taki sam, jak zwrócony przez poprzednie wywołanie MQCONN, ale z kodem zakończenia CCWARN i kodem przyczyny RC2002.

4. Gdy aplikacja zakończy się za pomocą wywołań IBM MQ , aplikacja powinna użyć wywołania MQDISC do rozłączenia się z menedżerem kolejek.
5. W systemie IBM i aplikacje napisane dla wersji wcześniejszych niż MQSeries V5.1 menedżera kolejek mogą być uruchamiane bez konieczności ponownego kompilowania.
6. To jest *tryb zgodności*. Ten tryb pracy udostępnia kompatybilne środowisko wykonawcze dla aplikacji napisanych przy użyciu dynamicznego łączenia. Składa się on z następujących elementów:
 - Program usługowy AMQZSTUB znajdujący się w bibliotece QMQM.
AMQZSTUB udostępnia ten sam interfejs publiczny co poprzednie wydania i ma ten sam podpis. Ten program usługowy może być używany do uzyskiwania dostępu do interfejsu MQI za pomocą wywołań procedur skonsolidowanych.
 - Program QMQM znajdujący się w bibliotece QMQM.
QMQM udostępnia sposób uzyskiwania dostępu do interfejsu MQI za pomocą dynamicznych wywołań programu.
 - Programy MQCLOSE, MQCONN, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQPUT1 i MQSET znajdujące się w bibliotece QMQM.
Programy te udostępniają również sposoby uzyskiwania dostępu do interfejsu MQI za pomocą dynamicznych wywołań programów, ale z listą parametrów, która odpowiada standardowi opisom wywołań IBM MQ .

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Te trzy interfejsy nie zawierają możliwości, które zostały wprowadzone w wersji 5.1. Na przykład wywołania MQBACK, MQCMIT i MQCONNX nie są obsługiwane. Obsługa udostępniana przez te interfejsy jest dostępna tylko dla aplikacji jednowątkowych.

Obsługa wywołań IBM MQ powiązanych statycznie w aplikacjach jednowątkowych, a także dla wszystkich wywołań IBM MQ w aplikacjach wielowątkowych, jest udostępniana za pośrednictwem programów usługowych LIBMQM i LIBMQM_R.

7. W systemie IBM i programy, które kończą się nieprawidłowo, nie są automatycznie odłączane od menedżera kolejek. Dlatego należy napisać aplikacje, aby zezwolić na możliwość wywołania MQCONN lub MQCONNX zwracającego kod zakończenia CCWARN i kod przyczyny RC2002. Uchwyt połączenia zwrócony w tej sytuacji może być używany jako normalny.

Parametry

Wywołanie MQCONN ma następujące parametry:

QMNAME (48-bajtowy łańcuch znaków)-wejście

Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, z którym aplikacja chce się połączyć. Nazwa może zawierać następujące znaki:

- Wielkie litery alfabetu (od A do Z)
- Małe litery alfabetu (od a do z)
- Cyfry cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani osadzonych odstępów, ale może zawierać odstępy końcowe. Znak o kodzie zero może być używany do wskazywania końca istotnych danych w nazwie; wartości null i dowolnych po nim znaków są traktowane jako znaki puste. W środowiskach wskazanych poniżej obowiązują następujące ograniczenia:

- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w znaki cudzysłowu, gdy są określone w komendach. Te znaki cudzysłowu nie mogą być określone w parametrze *QMNAME*.

Jeśli nazwa składa się całkowicie z odstępów, używana jest nazwa *domyślnego* menedżera kolejek.

Nazwa podana dla *QMNAME* musi być nazwą menedżera kolejek *connectable*.

Grupy współużytkowania kolejek: W systemach, w których istnieje kilka menedżerów kolejek i są one skonfigurowane w celu utworzenia grupy współużytkowania kolejki, nazwa grupy współużytkowania kolejki może zostać określona dla produktu *QMNAME* w miejsce nazwy menedżera kolejek. Umożliwia to aplikacji nawiązanie połączenia z *dowolnym* menedżerem kolejek, który jest dostępny w grupie współużytkowania kolejki. System można również skonfigurować w taki sposób, aby puste *QMNAME* powodował połączenie z grupą współużytkowania kolejki zamiast do domyślnego menedżera kolejek.

Jeśli parametr *QMNAME* określa nazwę grupy współużytkowania kolejki, ale istnieje również menedżer kolejek o tej nazwie w systemie, to połączenie jest nawiązane z tym drugim, preferowanym względem poprzedniego. Jeśli połączenie nie powiedzie się, to połączenie z jednym z menedżerów kolejek w grupie współużytkowania kolejki nie powiedzie się.

Jeśli połączenie powiedzie się, uchwyt zwrócony przez wywołanie *MQCONN* lub *MQCONN*X może być używany w celu uzyskania dostępu do *wszystkich* zasobów (współużytkowanych i niewspółużytkowanych) należących do konkretnego menedżera kolejek, do którego nawiązała połączenie. Dostęp do tych zasobów podlega typowym kontrolom autoryzacji.

Jeśli aplikacja wysyła dwa wywołania *MQCONN* lub *MQCONN*X w celu nawiązania współbieżnych połączeń, a jeden lub oba wywołania określają nazwę grupy współużytkowania kolejki, to drugie wywołanie może zwrócić kod zakończenia CCWARN i kod przyczyny RC2002. Dzieje się tak wtedy, gdy drugie wywołanie łączy się z tym samym menedżerem kolejek co pierwsze wywołanie.

Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS. Połączenie z grupą współużytkowania kolejki jest obsługiwane tylko w zadaniach wsadowych, wsadowych RRS i środowiskach TSO.

Aplikacje klienckie produktu IBM MQ: w przypadku aplikacji produktu IBM MQ MQI client połączenie jest wykonywane dla każdej definicji kanału połączenia klienckiego z określoną nazwą menedżera kolejek, dopóki nie powiedzie się to połączenie. Menedżer kolejek musi jednak mieć taką samą nazwę, jak określona nazwa. Jeśli zostanie podana pusta nazwa, każdy kanał połączenia klienckiego z pustą nazwą menedżera kolejek zostanie wypróbowany do czasu pomyślnego zakończenia. W takim przypadku nie będzie sprawdzania, czy nazwa menedżera kolejek jest rzeczywista.

Grupy menedżerów kolejek klienta produktu IBM MQ: Jeśli podana nazwa zaczyna się od gwiazdki (*), rzeczywisty menedżer kolejek, do którego nawiąże połączenie, może mieć nazwę inną niż nazwa określona przez aplikację. Podana nazwa (bez gwiazdki) definiuje *grupę* menedżerów kolejek, które kwalifikują się do połączenia. Implementacja wybiera jedną z grupy, próbując każdej z nich z kolei, w kolejności alfabetycznej, aż do momentu, w którym można będzie nawiązać połączenie. Jeśli żaden z menedżerów kolejek w grupie nie jest dostępny do połączenia, wywołanie nie powiedzie się. Każdy menedżer kolejek jest sprawdzany tylko raz. Jeśli dla nazwy została określona tylko gwiazdka, używana jest domyślna grupa menedżerów kolejek zdefiniowanych przez implementację.

Grupy menedżerów kolejek są obsługiwane tylko w przypadku aplikacji działających w środowisku klienta MQ. Wywołanie nie powiedzie się, jeśli aplikacja inna niż kliencka określa nazwę menedżera kolejek rozpoczynającą się od gwiazdki. Grupa jest definiowana przez udostępnienie kilku definicji kanału połączenia klienckiego z tą samą nazwą menedżera kolejek (określoną nazwą bez gwiazdki) w celu komunikowania się z każdym z menedżerów kolejek w grupie. Grupę domyślną definiuje się, podając co najmniej jedną definicję kanału połączenia klienckiego, każdy z pustą nazwą menedżera kolejek (podając wszystkie puste nazwy, w związku z tym ten sam efekt ma taki sam efekt, jak w przypadku podania jednej gwiazdki dla nazwy aplikacji klienckiej).

Po nawiązaniu połączenia z jednym menedżerem kolejek w grupie aplikacja może w typowy sposób określić odstępy w polach nazwy menedżera kolejek w deskryptorach komunikatów i obiektów, aby oznaczać nazwę menedżera kolejek, z którym aplikacja faktycznie nawiązała połączenie (*menedżer kolejek lokalnych*). Jeśli aplikacja musi znać tę nazwę, wywołanie MQINQ może zostać wysłane w celu sprawdzenia atrybutu menedżera kolejek produktu *QMqrName* .

Wstępne usunięcie gwiazdki z nazwą połączenia oznacza, że aplikacja nie jest zależna od łączenia się z określonym menedżerem kolejek w grupie. Odpowiednie aplikacje byłyby następujące:

- Aplikacje, które umieszczają komunikaty, ale nie dostają komunikatów.
- Aplikacje, które umieszczają komunikaty żądań, a następnie otrzymują komunikaty odpowiedzi z kolejki *tymczasowej dynamicznej* .

Nieodpowiednie aplikacje to te, które muszą pobrać komunikaty z określonej kolejki w określonym menedżerze kolejek. Takie aplikacje nie powinny prefikować nazwy za pomocą gwiazdki.

Należy zwrócić uwagę, że jeśli zostanie podana gwiazdka, maksymalna długość pozostałej części nazwy to 47 znaków.

Długość tego parametru jest podawana przez LNQMNM.

HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Musi być ona określona we wszystkich kolejnych wywołaniach kolejkowania komunikatów wywołanych przez aplikację. Traci ona ważność po wywołaniu wywołania MQDISC lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, kończy działanie.

Zakres uchwytu jest ograniczony do najmniejszej jednostki przetwarzania równoległe obsługiwane przez platformę, na której działa aplikacja; uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołano wywołanie MQCONN.

- W systemie IBM zakres uchwytu jest zadaniem wydającym wywołanie.

W systemie IBM i dla aplikacji działających w trybie zgodności zwracana jest wartość:

HCDEFH

Domyślny uchwyt połączenia.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2002

(2002, X'7D2') Aplikacja jest już połączona.

Jeśli *CMPCOD* to *CCFAIL*:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2267

(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2035

(2035, X'7F3') Brak uprawnień do dostępu.

RC2137

(2137, X'859 ') Obiekt nie został otwarty pomyślnie.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2063

(2063, X'80F') Wystąpił błąd zabezpieczeń.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C          CALLP      MQCONN(QMNAME : HCONN : CMPCOD :  
C                      REASON)
```

Definicja prototypu dla wywołania to:

```
D*.1.....2.....3.....4.....5.....6.....7..  
DMQCONN      PR          EXTPROC('MQCONN')  
D* Name of queue manager  
D QMNAME          48A  
D* Connection handle  
D HCONN          10I 0  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CMPCOD  
D REASON          10I 0
```

MQCONNX-Connect menedżer kolejek (rozszerzony)

Wywołanie MQCONNX łączy program aplikacji z menedżerem kolejek. Udostępnia uchwyt połączenia menedżera kolejek, który jest używany przez aplikację przy kolejnych wywołaniach programu IBM MQ .

Wywołanie MQCONN_X jest podobne do wywołania MQCONN, z tą różnicą, że MQCONN_X umożliwia określenie opcji sterujących sposobem, w jaki działa wywołanie.

- W systemie IBM iwywołanie to nie jest obsługiwane w przypadku aplikacji działających w trybie zgodności.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

W systemach IBM MQ for Windows, UNIX i IBM i każdy wątek w aplikacji może łączyć się z różnymi menedżerami kolejek. W innych systemach wszystkie współbieżne połączenia w ramach procesu muszą należeć do tego samego menedżera kolejek.

- [“Składnia” na stronie 1303](#)
- [“Parametry” na stronie 1303](#)
- [“Deklaracja RPG” na stronie 1304](#)

Składnia

MQCONN_X (*QMNAME*, *CNOPT*, *HCONN*, *CMPCOD*, *REASON*)

Parametry

Wywołanie MQCONN_X ma następujące parametry:

QMNAME (48-bajtowy łańcuch znaków)-wejście

Nazwa menedżera kolejek.

Szczegółowe informacje zawiera opis parametru *QMNAME* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek” na stronie 1298](#).

CNOPT (MQCNO)-wejście/wyjście

Opcje, które sterują działaniem MQCONN_X.

Szczegółowe informacje na ten temat zawiera sekcja [“MQCNO-opcje połączenia” na stronie 1062](#).

HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt połączenia.

Szczegółowe informacje zawiera opis parametru *HCONN* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek” na stronie 1298](#).

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Szczegółowe informacje zawiera opis parametru *CMPCOD* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek” na stronie 1298](#).

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Szczegółowe informacje na temat możliwych kodów przyczyny zawiera opis parametru *REASON* opisany w sekcji [“MQCONN-Połączenie menedżera kolejek” na stronie 1298](#).

Wywołanie MQCONN_X może zwrócić następujące dodatkowe kody przyczyny:

Jeśli *CMPCOD* to CCFAIL:

RC2278

(2278, X'8E6') Pola połączenia klienta nie są poprawne.

RC2139

(2139, X'85B') Struktura opcji Connect nie jest poprawna.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                               CALLP      MQCONN(QMNAME : HCONN : CMPCOD :  
C                               REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQCONN          PR          EXTPROC('MQCONN')  
D* Name of queue manager  
D QMNAME          48A  
D* Options that control the action of MQCONN  
D HCONN          224A  
D* Connection handle  
D HCONN          10I 0  
D* Completion code  
D CMPCOD          10I 0  
D* Reason code qualifying CMPCOD  
D REASON          10I 0
```

MQCRTMH-Tworzenie uchwytu komunikatu

Wywołanie MQCRTMH zwraca uchwyt komunikatu.

Aplikacja może używać jej w kolejnych wywołaniach kolejowania komunikatów:

- Użyj wywołania [MQSETMP](#) , aby ustawić właściwość uchwytu komunikatu.
- Użyj wywołania [MQINQMP](#) , aby dowiedzieć się o wartości właściwości uchwytu komunikatu.
- Za pomocą wywołania [MQDLTMP](#) można usunąć właściwość uchwytu komunikatu.

Uchwyt komunikatu może być używany w wywołaniach MQPUT i MQPUT1 , aby powiązać właściwości uchwytu komunikatu z właściwościami umieszczanego komunikatu. Podobnie, określając uchwyt komunikatu w wywołaniu MQGET, właściwości pobieranego komunikatu można uzyskać za pomocą uchwytu komunikatu po zakończeniu wywołania MQGET.

Użyj komendy [MQDLTMH](#) , aby usunąć uchwyt komunikatu.

- [“Składnia” na stronie 1304](#)
- [“Parametry” na stronie 1304](#)
- [“Deklaracja RPG” na stronie 1306](#)

Składnia

MQCRTMH (*Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason*)

Parametry

Wywołanie MQCRTMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. Jeśli połączenie z menedżerem kolejek przestanie być poprawne, a w uchwycie komunikatu nie działa żaden wywołanie produktu IBM MQ , komenda [MQDLTMH](#) jest niejawnie wywoływana w celu usunięcia komunikatu.

Alternatywnie można podać następującą wartość:

HCUNAS

Uchwyty połączenia nie reprezentuje połączenia z żadnym określonym menedżerem kolejek.

Jeśli ta wartość jest używana, uchwyt komunikatu musi zostać usunięty z jawnym wywołaniem komendy MQDLTMH w celu zwolnienia przydzielonej pamięci masowej; IBM MQ nigdy niejawnie usuwa uchwyt komunikatu.

Musi istnieć co najmniej jedno poprawne połączenie z menedżerem kolejek ustanowionym w wątku tworzący uchwyt komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd RC2018.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także określić następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja Tryb kompatybilności dla aplikacji RPG w systemie IBM i

CRTOPT (MQCMHO)-dane wejściowe

Opcje, które sterują działaniem MQCRTMH. Szczegółowe informacje na ten temat zawiera sekcja MQCMHO .

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wyjście

Na wyjściu zwracany jest uchwyt komunikatu, który może być używany do ustawiania, sprawdzania i usuwania właściwości uchwytu komunikatu. Początkowo uchwyt komunikatu nie zawiera żadnych właściwości.

Uchwyt komunikatu ma również powiązany deskryptor komunikatu. Początkowo ten deskryptor komunikatu zawiera wartości domyślne. Wartości powiązanych pól deskryptora komunikatu można ustawić i dowiedzieć się, używając wywołań MQSETMP i MQINQMP. Wywołanie MQDLTMP resetuje pole deskryptora komunikatu z powrotem do jego wartości domyślnej.

Jeśli parametr *HCONN* jest określony jako wartość HCUNAS, to zwrócony uchwyt komunikatu może być używany w wywołaniach MQGET, MQPUT lub MQPUT1 z dowolnym połączeniem w jednostce przetwarzania, ale może być używany tylko przez jedno wywołanie produktu IBM MQ naraz. Jeśli uchwyt jest używany, gdy druga próba wywołania IBM MQ próbuje użyć tego samego uchwytu komunikatu, drugie wywołanie IBM MQ nie powiedzie się i zostanie zwrócony kod przyczyny RC2499.

Jeśli parametr *HCONN* nie jest parametrem HCUNAS, to zwrócony uchwyt komunikatu może być używany tylko w określonym połączeniu.

Ta sama wartość parametru *HCONN* musi być używana w kolejnych wywołaniach MQI, w których ten uchwyt komunikatu jest używany:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

Zwrócony uchwyt komunikatu przestaje być poprawny, gdy dla uchwytu komunikatu zostanie wywołane wywołanie MQDLTMH lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostanie zakończona. Komenda MQDLTMH jest wywoływana niejawnie, jeśli podczas tworzenia uchwytu komunikatu określone jest konkretne połączenie, a połączenie z menedżerem kolejek przestaje być poprawne, na przykład jeśli wywołano komendę MQDBC.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2461

(2461, X'099D') Struktura opcji uchwytu komunikatu nie jest poprawna.

RC2273

(2273, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2017

(2017, X'07E1') Nie ma więcej dostępnych uchwytów.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2460

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej szczegółów na ten temat zawiera sekcja [“Kody powrotu dla IBM i \(ILE RPG\)”](#) na stronie 1455.

Deklaracja RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQCRTMH(HCONN : CRTOPT : HMSG :  
                                CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
DMQCRTMH      PR      EXTPROC('MQCRTMH')  
D* Connection handle
```

D HCONN	10I 0 VALUE
D* Options that control the action of MQCRTMH	
D CRTOPT	12A
D* Message handle	
D HMSG	20I 0
D* Completion code	
D CMPCOD	10I 0
D* Reason code qualifying CompCode	
D REASON	10I 0

MQCTL-wywołanie zwrotne elementu sterującego

Wywołanie MQCTL wykonuje operacje sterujące dla uchwytów obiektów otwartych dla połączenia.

- [“Składnia” na stronie 1307](#)
- [“Użycie notatek” na stronie 1307](#)
- [“Parametry” na stronie 1307](#)
- [“Deklaracja RPG” na stronie 1312](#)

Składnia

MQCTL (*Hconn, Operation, ControlOpts, CompCode, Reason*)

Użycie notatek

1. Procedury wywołania zwrotnego muszą sprawdzać odpowiedzi ze wszystkich wywołanych przez nie usług, a jeśli procedura wykryje warunek, którego nie można rozwiązać, musi wywołać komendę MQCB (CBREG), aby zapobiec powtórnym wywoławczym wywołaniom procedury zwrotnej.

Parametry

Wywołanie MQCTL ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a ponadto można określić następującą wartość specjalną dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

OPERATN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Operacja jest przetwarzana dla wywołania zwrotnego zdefiniowanego dla podanego uchwytu obiektu. Należy określić jedną i tylko jedną z następujących opcji:

CTLSR

Uruchamia konsumowanie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Wywołania zwrotne są uruchamiane w wątku uruchomionym przez system, który różni się od żadnego z wątków aplikacji.

Ta operacja umożliwia sterowanie udostępnionym uchwytym połączenia z systemem. Jedynymi wywołaniami MQI, które mogą być wysyłane przez wątek inny niż wątek konsumenta, są:

- MQCTL z operacją CTLSR

- MQCTL z operacją CTLSU
- MQDISC-wykonuje operację MQCTL z operacją CTLSP przed rozłączonym połączeniem z HConn.

Wartość RC2500 jest zwracana, jeśli wywołanie funkcji API produktu IBM MQ zostanie wysłane podczas uruchamiania uchwytu połączenia, a wywołanie nie pochodzi z funkcji konsumenta komunikatów.

Jeśli połączenie nie powiedzie się, ta konwersacja zostanie zatrzymana tak szybko, jak to możliwe. Możliwe jest zatem, aby wywołanie funkcji API IBM MQ w głównym wątku odebrało kod powrotu RC2500 na chwilę, a następnie kod powrotu RC2009, gdy połączenie powróci do stanu zatrzymania.

Może to być wydane w funkcji konsumenta. Dla tego samego połączenia, co procedura zwrotna, jej jedynym celem jest anulowanie poprzednio wystawionej operacji CTLSP.

Ta opcja nie jest obsługiwana, jeśli aplikacja jest powiązana z niewielowątkową biblioteką IBM MQ.

CTLSW

Uruchamia konsumowanie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Odbiorcy komunikatów działają w tym samym wątku, a sterowanie nie jest zwracane do programu wywołującego MQCTL, dopóki:

- Wydane przez użycie operacji MQCTL CTLSP lub CTLSU, lub
- Wszystkie procedury konsumenckie zostały wyrejestrowywane lub zawieszone.

Jeśli wszyscy konsumenci zostaną wyrejestrowani lub zawieszeni, zostanie wykonana niejawna operacja CTLSP.

Ta opcja nie może być używana w ramach procedury zwrotnej ani dla bieżącego uchwytu połączenia, ani dla żadnego innego uchwytu połączenia. Jeśli wywołanie zostanie uruchomione, zostanie zwrócone z RC2012.

Jeśli w dowolnym momencie w trakcie operacji CTLSW nie ma zarejestrowanych, niezawieszonych konsumentów wywołanie nie powiedzie się, kod przyczyny RC2446.

Jeśli podczas wykonywania operacji CTLSW połączenie zostanie zawieszone, wywołanie MQCTL zwróci kod przyczyny ostrzeżenia RC2521; , połączenie pozostanie "uruchomione".

Aplikacja może zdecydować o wydaniu CTLSP lub CTLRE. W tym przypadku bloki operacji CTLRE.

Ta opcja nie jest obsługiwana w przypadku pojedynczego klienta wielowątkowego.

CTLSP

Przed zakończeniem tej opcji zatrzymaj korzystanie z komunikatów i poczekaj na zakończenie operacji przez wszystkich konsumentów. Ta operacja zwalnia uchwyt połączenia.

Jeśli ta opcja jest wydawana w ramach procedury zwrotnej, ta opcja nie jest uwzględniana do czasu zakończenia procedury. Po zakończeniu procedur konsumenckich dla komunikatów, które zostały już odczytane, nie są wywoływane żadne procedury dla konsumenta komunikatów. Po zatrzymaniu wywołań zwrotnych (jeśli jest to wymagane) do procedur zwrotnych, zostaną wykonane procedury.

Jeśli jest ona wystawiona poza procedurą wywołania zwrotnego, sterowanie nie jest zwracane do programu wywołującego, dopóki nie zostaną wykonane procedury konsumenckie dla komunikatów, które zostały już odczytane, i po zatrzymaniu wywołań zwrotnych (jeśli jest wymagane) do wywołań zwrotnych. Jednak same procedury zwrotne pozostają zarejestrowane.

Ta funkcja nie ma wpływu na komunikaty odczytu z wyprzedzeniem. Należy upewnić się, że konsumenci uruchamiają komendę MQCLOSE (COQSC), z poziomu funkcji zwrotnej, w celu określenia, czy są dostępne dalsze komunikaty, które można dostarczyć.

CTLSU

Wstrzymaj korzystanie z komunikatów. Ta operacja zwalnia uchwyt połączenia.

Nie wpływa to na odczyt z wyprzedzeniem komunikatów dla aplikacji. Jeśli komunikaty mają zostać zatrzymane przez długi czas, należy rozważyć zamknięcie kolejki i ponowne jej otwarcie, gdy należy kontynuować korzystanie z tej kolejki.

Jeśli jest ona wydana z poziomu procedury zwrotnej, nie jest ona skuteczna, dopóki procedura nie zostanie zakończona. Po zakończeniu bieżących procedur zewnętrznych nie będą wywoływane żadne procedury konsumenta komunikatów.

Jeśli zostanie ona wydana poza wywołaniem zwrotnym, sterowanie nie zostanie zwrócone do programu wywołującego, dopóki nie zostanie zakończona bieżąca procedura konsumenta i nie zostanie już wywołana żadna wartość.

CTLRE

Wznów korzystanie z komunikatów.

Ta opcja jest zwykle wydawana z głównego wątku aplikacji, ale może być również używana w ramach procedury zwrotnej w celu anulowania wcześniejszego żądania zawieszenia wydanego w tej samej procedurze.

Jeśli CTLRE zostanie użyte do wznowienia CTLSW, to bloki operacji.

PCTLOP (MQCTLO)-dane wejściowe

Opcje sterujące działaniem komendy MQCTL

Szczegółowe informacje na temat struktury można znaleźć w sekcji [MQCTLO](#).

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Następujące kody przyczyn to te, które menedżer kolejek może zwrócić dla parametru *Reason*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFail:

RC2133

(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

RC2204

(2204, X'89C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2374

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się.

RC2183

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2005

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

RC2487

(2487, X'9B7') Nie można wywołać procedury zwrotnej

RC2448

(2448, X' 990 ') Nie można wyrejestrować, zawiesić ani wznowić, ponieważ nie ma zarejestrowanej procedury zwrotnej.

RC2486

(2486, X'9B6') Albo zarówno CallbackFunction , jak i CallbackName zostały określone w wywołaniu CBREG, albo albo jeden z CallbackFunction , albo CallbackName został określony, ale nie jest zgodny z aktualnie zarejestrowaną funkcją zwrotną.

RC2483

(2483, X'9B3') Niepoprawne pole typu CallBackType.

RC2219

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2444

(2444, X'98C') Blok opcji jest niepoprawny.

RC2484

(2484, X'9B4') Niepoprawne pole opcji MQCBD.

RC2140

(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez produkt CICS.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2217

(2217, X'8A9') Brak uprawnień do połączenia.

RC2202

(2202, X'89A') Połączenie wygaszające.

RC2203

(2203, X'89B') Połączenie jest zamykane.

RC2207

(2207, X'89F') Błąd korelacji identyfikatora.

RC2016

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

RC2351

(2351, X'92F') Global units of work conflict (Globalne jednostki pracy w konflikcie).

RC2186

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

RC2353

(2353, X' 931 ') Uchwyt w użyciu dla globalnej jednostki pracy.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2259

(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2352

(2352, X' 930 ') Global unit of work conflicts with local unit of work.

RC2247

(2247, X'8C7') Opcje zgodności nie są poprawne.

RC2485

(2485, X'9B5') Niepoprawne pole długości MaxMsg

RC2026

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

RC2497

(2497, X'9C1') Określony punkt wejścia funkcji nie został znaleziony w module.

RC2496

(2496, X'9C0') Moduł został znaleziony, ale ma niepoprawny typ (32-lub 64-bitowy) lub nie jest poprawną biblioteką dll.

RC2495

(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie został autoryzowany do załadowania.

RC2206

(2206, X'89E') Błąd identyfikatora komunikatu.

RC2250

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

RC2331

(2331, X'91B') Użycie znacznika komunikatu nie jest poprawne.

RC2036

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2488

(2488, X'9B8') Niepoprawny kod operacji w wywołaniu API Call

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2193

(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2394

(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2069

(2069, X'815 ') Signal outstanding for this handle.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2109

(2109, X'83D') Wywołanie zostało pominięte przez program obsługi wyjścia.

RC2072

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2354

(2354, X' 932 ') Enlistment in global unit of work failed (Enlistment in global unit of work failed).

RC2355

(2355, X' 933 ') Mixture of unit-of-work calls not supported.

RC2255

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

RC2090

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

RC2256

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

RC2257

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

RC2298

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQCTL(HCONN : OPERATN : PCTLOP :
                                           CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

DMQCTL          PR          EXTPROC('MQCTL')
D* Connection handle
D HCONN          10I 0 VALUE
D* Operation
D OPERATN        10I 0 VALUE
D* Control options
D PCTLOP          32A
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CompCode
D REASON         10I 0

```

MQDISC-rozłączenie menedżera kolejek

Wywołanie MQDISC przerywa połączenie między menedżerem kolejek a programem użytkowym. Jest to odwrotność wywołania MQCONN lub MQCONNX.

- W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Więcej informacji zawiera temat [“MQCONN-Połączenie menedżera kolejek”](#) na stronie 1298.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

- [“Składnia”](#) na stronie 1313

- [“Użycie notatek” na stronie 1313](#)
- [“Parametry” na stronie 1313](#)
- [“Deklaracja RPG” na stronie 1314](#)

Składnia

MQDISC (*HCONN*, *CMPCOD*, *REASON*)

Użycie notatek

1. Jeśli wywołanie MQDISC jest wysyłane, gdy aplikacja nadal ma otwarte obiekty, te obiekty są zamykane przez menedżer kolejek, a opcje zamknięcia są ustawione na wartość CONONE.
2. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, dyspozycja tych zmian zależy od tego, w jaki sposób aplikacja kończy pracę:
 - a. Jeśli aplikacja wysyła wywołanie MQDISC przed zakończeniem działania:
 - W przypadku koordynowanej jednostki pracy menedżera kolejek menedżer kolejek wysyła wywołanie MQCMIT w imieniu aplikacji. Jeśli jest to możliwe, jednostka pracy jest zatwierdzana i wycofana, jeśli nie jest dostępna.
 - W przypadku zewnętrznio koordynowanej jednostki pracy nie ma zmian w statusie jednostki pracy; jednak menedżer kolejek wskaże, że jednostka pracy powinna zostać zatwierdzona przez koordynatora jednostki pracy.
 - b. Jeśli aplikacja zakończy się normalnie, ale bez wywoływania wywołania MQDISC, tworzona jest kopia zapasowa jednostki pracy.
 - c. Jeśli aplikacja zakończy działanie *nieprawidłowo* bez wywoływania wywołania MQDISC, wycofana jest jednostka pracy.
3. W systemie IBM i aplikacje działające w trybie zgodności nie muszą wywoływać tego wywołania. Więcej informacji można znaleźć w wywołaniu MQCONN.

Parametry

Wywołanie MQDISC ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN*:

HCDEFH

Domyślny uchwyt połączenia.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia wartość *HCONN* na wartość, która nie jest poprawnym uchwytem dla środowiska. Ta wartość jest następująca:

HCUNUH

Uchwyt połączenia bez użycia.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQDISC(HCONN : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQDISC                                PR      EXTPROC('MQDISC')
D* Connection handle
D HCONN                                10I 0
D* Completion code
D CMPCOD                                10I 0
D* Reason code qualifying CMPCOD
D REASON                                10I 0
```

MQDLTMH-usuwanie uchwytu komunikatu

Wywołanie MQDLTMH usuwa uchwyt komunikatu i jest odwrotnym wywołaniem wywołania MQCRTMH.

- [“Składnia” na stronie 1315](#)
- [“Użycie notatek” na stronie 1315](#)
- [“Parametry” na stronie 1316](#)

- [“Deklaracja RPG” na stronie 1318](#)

Składnia

MQDLTMH ((*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Reason*))

Użycie notatek

1. Tego wywołania można użyć tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Może to być:

- Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
- Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .

Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja [“MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 1274](#).

2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiednich wywołań zwrotnych zamiast MQBACK. Środowisko może również obsługiwać niejawne wycofania spowodowane przez nieprawidłowe zakończenie działania aplikacji.
 - W systemie z/OS należy użyć następujących wywołań:
 - Programy wsadowe (w tym wsadowe programy DL/I produktu IMS) mogą używać wywołania MQBACK, jeśli jednostka pracy ma wpływ tylko na zasoby IBM MQ . Jeśli jednak jednostka pracy ma wpływ na zasoby zarówno IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnionego przez usługę RRS (Recoverable Resource Service) produktu z/OS . Wywołanie SRRBACK powoduje wycofania zmian w zasobach należących do menedżerów zasobów, które zostały włączone dla koordynacji RRS.
 - Aplikacje produktu CICS muszą używać komendy EXEC CICS SYNCPOINT ROLLBACK do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
 - Aplikacje produktu IMS (inne niż wsadowe programy DL/I) muszą używać wywołań programu IMS , takich jak ROLB , do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż wsadowe programy DL/I).
 - W systemie IBM należy użyć tego wywołania dla lokalnych jednostek pracy koordynowanych przez menedżera kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE (*JOB) nie może zostać wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcji [“MQDISC-rozłączenie menedżera kolejek” na stronie 1312](#) .
4. Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwytym kolejki i obejmują takie elementy jak:
 - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset* i *MsgFlags* w strukturze MQMD.
 - Określa, czy komunikat jest częścią jednostki pracy.
 - W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje trzy zestawy informacji o grupach i segmentach, jeden zestaw dla każdego z następujących elementów:

 - Ostatnie pomyślne wywołanie MQPUT (może to być część jednostki pracy).
 - Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).

- Ostatnie pomyślne wywołanie MQGET, które przeglądało komunikat w kolejce (nie może to być część jednostki pracy).

Jeśli aplikacja wstawi lub pobiera komunikaty jako część jednostki pracy, a następnie aplikacja utworzy kopię zapasową jednostki pracy, wówczas informacje o grupie i segmencie zostaną odtworzone do wartości, którą poprzednio:

- Informacje powiązane z wywołaniem MQPUT są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
- Informacje powiązane z wywołaniem MQGET są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zakresem jednostki pracy, nie mają informacji o grupach i segmentach, które zostały odtworzone, jeśli zostanie utworzona kopia zapasowa jednostki pracy.

Odtwarzanie informacji o grupach i segmentach do jej poprzedniej wartości, gdy tworzona jest kopia zapasowa jednostki pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy, a także zrestartowanie w poprawnym punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczające informacje, aby móc restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu.

Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu można znaleźć w sekcji PMLOGO opisanej w sekcji [PMOPT \(10-cyfrowa liczba podpisanych cyfr\)](#), a także w opcji GMLOGO opisanej w sekcji [GMOPT \(10-cyfrowa liczba całkowita ze znakiem\)](#).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

5. Jednostka pracy ma ten sam zasięg co uchwyt połączenia. Wszystkie wywołania produktu IBM MQ, które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń znajdują się w sekcji [HCONN \(10-cyfrowa liczba całkowita ze znakiem\)-dane wyjściowe](#).
6. To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.
7. Długo działająca aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapełnić kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć tę możliwość, administrator musi ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapełnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duże, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQDLTMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *HMSG*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy HCUNAS, konieczne jest nawiązanie poprawnego połączenia w wątku usuwaniu uchwytu komunikatu, w przeciwnym razie wywołanie nie powiedzie RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-input/output

To jest uchwyt komunikatu, który ma zostać usunięty. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

Po pomyślnym zakończeniu wywołania uchwyt jest ustawiany na niepoprawną wartość dla środowiska. Ta wartość jest następująca:

HMUNUH

Uchwyt komunikatu nie do użycia.

Uchwyt komunikatu nie może zostać usunięty, jeśli w toku jest inne wywołanie produktu IBM MQ , które zostało przekazane temu samemu uchwycie komunikatu.

DLTOPT (MQDMHO)-dane wejściowe

Szczegółowe informacje na ten temat zawiera sekcja [MQDMHO](#) .

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CMPCOD* ma wartość CCFail:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2462

(2462, X'099E') Struktura opcji uchwytu komunikatu usuwania nie jest poprawna.

RC2460

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej szczegółów na ten temat zawiera sekcja [“Kody powrotu dla IBM i \(ILE RPG\)”](#) na stronie 1455.

Deklaracja RPG

```
C*.1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQDLTMH(HCONN : HMSG : DLTOPT :  
                                CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
DMQDLTMH          PR              EXTPROC('MQDLTMH')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Message handle  
D HMSG              20I 0  
D* Options that control the action of MQDLTMH  
D DLTOPT            12A  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CompCode  
D REASON            10I 0
```

MQDLTMP-usunięcie właściwości komunikatu

Wywołanie MQDLTMP usuwa właściwość z uchwytu komunikatu i jest odwrotną wartością wywołania MQSETMP.

- [“Składnia” na stronie 1318](#)
- [“Parametry” na stronie 1318](#)
- [“Deklaracja RPG” na stronie 1320](#)

Składnia

MQDLTMP (*Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason*)

Parametry

Wywołanie MQDLTMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-Wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość musi być zgodna z uchwytym połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *HMSG*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy HCUNAS, należy ustanowić poprawne połączenie w wątku usuwaniu uchwytu komunikatu, w przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Jest to uchwyt komunikatu zawierający właściwość do usunięcia. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

DLTOPT (MQDMPO)-Dane wejściowe

Szczegółowe informacje zawiera opis typu danych [MQDMPO](#).

PRNAME (MQCHARV)-dane wejściowe

Nazwa właściwości do usunięcia. Więcej informacji na temat nazw właściwości zawiera sekcja [Nazwy właściwości](#).

Znaki wieloznaczne nie są dozwolone w nazwie właściwości.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2471

(2471, X'09A7') Właściwość nie jest dostępna.

RC2421

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CMPCOD* to CCFAIL:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'086D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2481

(2481, X'09B1') Struktura opcji usuwania właściwości komunikatu nie jest poprawna.

RC2460

(2460, X'099C') Uchwyty komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2442

(2442, X'098A') Niepoprawna nazwa właściwości.

RC2111

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

RC2195

(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat tych kodów zawiera sekcja [Kody zakończenia i kody przyczyny interfejsu API](#).

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQDLTMP(HCONN : HMSG : DLTOPT :  
                                      PRNAME : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
DMQDLTMP          PR          EXTPROC('MQDLTMP')  
D* Connection handle  
D HCONN           10I 0 VALUE  
D* Message handle  
D HMSG           20I 0 VALUE  
D* Options that control the action of MQDLTMP  
D DLTOPT         12A  
D* Property name  
D PRNAME         32A  
D* Completion code  
D CMPCOD         10I 0  
D* Reason code qualifying CompCode  
D REASON         10I 0
```

MQGET-Pobieranie komunikatu

Wywołanie MQGET pobiera komunikat z kolejki lokalnej, który został otwarty za pomocą wywołania MQOPEN.

- [“Składnia” na stronie 1320](#)
- [“Użycie notatek” na stronie 1320](#)
- [“Parametry” na stronie 1323](#)
- [“Deklaracja RPG” na stronie 1328](#)

Składnia

MQGET (*HCONN*, *HOBJ*, *MSGDSC*, *GMO*, *BUFLEN*, *BUFFER*, *DATLEN*, *CMPCOD*, *REASON*)

Użycie notatek

1. Pobrany komunikat jest zwykle usuwany z kolejki. To usunięcie może zostać wykonane jako część wywołania MQGET lub jako część punktu synchronizacji. Usunięcie komunikatu nie występuje, jeśli w parametrze *GMO* podano opcję GMBRWF lub GMBRWN (patrz pole *GMOPT* opisane w sekcji [“MQGMO-Opcje Get-message” na stronie 1089](#)).
2. Jeśli opcja GMLK jest określona z jedną z opcji przeglądania, przejrzany komunikat jest zablokowany w taki sposób, że jest widoczny tylko dla tego uchwytu.

Jeśli zostanie podana opcja GMUNLK, to poprzednio zablokowany komunikat zostanie odblokowany. W tym przypadku żaden komunikat nie jest pobierany, a parametry *MSGDSC*, *BUFLEN*, *BUFFER* i *DATLEN* nie są sprawdzane ani zmieniane.
3. Jeśli aplikacja wywołująca wywołanie MQGET działa jako IBM MQ MQI client, to możliwe jest, że komunikat zostanie utracony, jeśli podczas przetwarzania wywołania MQGET IBM MQ MQI client zostanie zakończone nieprawidłowo lub połączenie klienta zostanie zerwane. Wynika to z faktu, że odpowiedniki działające na platformie menedżera kolejek i które wywołują wywołanie MQGET w imieniu klienta, nie mogą wykryć utraty klienta do momentu, w którym surogat nie zwróci komunikatu do klienta. Jest to po usunięciu komunikatu z kolejki. Może to wystąpić zarówno w przypadku trwałych komunikatów, jak i nietrwałych komunikatów.

Ryzyko utraty wiadomości w ten sposób może zostać wyeliminowane przez zawsze pobieranie komunikatów w jednostkach pracy (czyli poprzez określenie opcji GMSYP w wywołaniu MQGET i użycie wywołania MQCMIT lub MQBACK w celu zatwierdzenia lub wycofania jednostki pracy

po zakończeniu przetwarzania komunikatu). Jeśli określono GMSYP, a klient zakończy działanie w sposób nieprawidłowy lub połączenie zostanie zerwane, zastępcze wycofają jednostkę pracy w menedżerze kolejek i komunikat zostanie przywrócony do kolejki.

W zasadzie ta sama sytuacja może powstać w przypadku aplikacji działających na platformie menedżera kolejek, ale w tym przypadku okno, w którym może zostać utracony komunikat, jest niewielkie. Jednak, podobnie jak w przypadku produktu IBM MQ MQI clients, ryzyko może zostać wyeliminowane przez pobranie komunikatu w ramach jednostki pracy.

4. Jeśli aplikacja umieszcza sekwencję komunikatów w konkretnym przypadku kolejce w ramach pojedynczej jednostki pracy, a następnie zatwierdza tę jednostkę pracy pomyślnie, komunikaty stają się dostępne do pobrania w następujący sposób:
 - Jeśli kolejka jest kolejką *niewspółużytkowaną* (czyli kolejką lokalną), wszystkie komunikaty w obrębie jednostki pracy stają się dostępne w tym samym czasie.
 - Jeśli kolejka jest kolejką *współużytkowaną*, komunikaty w obrębie jednostki pracy stają się dostępne w kolejności, w jakiej zostały umieszczone, ale nie wszystkie w tym samym czasie. Jeśli system jest mocno obciążony, to jest możliwe, aby pierwszy komunikat w jednostce pracy został pomyślnie pobrany, ale wywołanie MQGET dla drugiego lub kolejnego komunikatu w jednostce pracy nie powiedzie się i zostanie zwrócony błąd RC2033. W takim przypadku aplikacja musi czekać na krótką chwilę, a następnie ponowić operację.
5. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, kolejność tych komunikatów jest zachowywana, jeśli spełnione są określone warunki. Szczegółowe informacje można znaleźć w uwagach dotyczących użycia w opisie wywołania MQPUT. Jeśli warunki są spełnione, komunikaty są prezentowane w aplikacji odbierającej w kolejności, w jakiej zostały wysłane, jeżeli:
 - Tylko jeden odbiorca otrzymuje komunikaty z kolejki.

Jeśli istnieją dwie lub więcej aplikacji pobierających komunikaty z kolejki, muszą one uzgodnić z nadawcą mechanizm używany do identyfikowania komunikatów należących do sekwencji. Na przykład nadawca może ustawić wszystkie pola *MDCID* w komunikatach w sekwencji do wartości, która była unikalna dla tej sekwencji komunikatów.
 - Odbiornik nie zmienia celowo kolejności pobierania, na przykład przez określenie konkretnej *MDMID* lub *MDCID*.

Jeśli aplikacja wysyłający przekaże komunikaty jako grupę komunikatów, komunikaty są prezentowane w aplikacji odbierającej w poprawnej kolejności, jeśli aplikacja odbierająca określi opcję GMLOGO w wywołaniu MQGET. Więcej informacji na temat grup komunikatów zawiera sekcja:

 - Pole *MDMFL* w strukturze MQMD
 - Opcja PMLOGO w MQPMO
 - Opcja GMLOGO w MQGMO
6. Test aplikacji dla kodu sprzężenia zwrotnego FBQUIT w polu *MDFB* w parametrze *MSGDSC*. Jeśli ta wartość zostanie znaleziona, aplikacja kończy działanie. Więcej informacji na ten temat zawiera sekcja *MDFB* opisana w sekcji [“MQMD-deskryptor komunikatu”](#) na stronie 1124.
7. Jeśli kolejka identyfikowana przez produkt *HOB*J została otwarta za pomocą opcji OOSAVA, a kod zakończenia z wywołania MQGET to CCOK lub CCWARN, kontekst powiązany z uchwyceniem kolejki *HOB*J jest ustawiany na kontekst komunikatu, który został pobrany (chyba że ustawiona jest opcja GMBRWF lub GMBRWN, w takim przypadku kontekst jest oznaczony jako niedostępny). Tego kontekstu można użyć w kolejnych wywołaniach MQPUT lub MQPUT1, podając opcje PMPASI lub PMPASA. Umożliwia to przesyłanie kontekstu komunikatu, który ma zostać przesłany w całości lub w części, do innego komunikatu (na przykład, gdy komunikat jest przekazywany do innej kolejki). Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).
8. Jeśli opcja GMCONV jest uwzględniona w parametrze *GMO*, dane komunikatu aplikacji są przekształcane na reprezentację żądaną przez aplikację odbierającą, zanim dane zostaną umieszczone w parametrze *BUFFER*:

- Pole *MDFMT* znajdujące się w informacjach sterujących w komunikacie identyfikuje strukturę danych aplikacji, a pola *MDCSI* i *MDENC* w informacjach sterujących w komunikacie określają jego identyfikator i kodowanie zestawu znaków.
- Aplikacja wywołująca wywołanie MQGET określa w polach *MDCSI* i *MDENC* w parametrze *MSGDSC* identyfikator zestawu znaków i kodowanie, do którego muszą zostać przekształcone dane komunikatu aplikacji.

Gdy konieczna jest konwersja danych komunikatu, konwersja jest wykonywana przez sam menedżer kolejek lub przez wyjście napisane przez użytkownika, w zależności od wartości pola *MDFMT* w informacjach sterujących w komunikacie:

- Następujące formaty są automatycznie przekształcane przez menedżera kolejek. Te formaty są nazywane formatami wbudowanymi:

FMADMN	FMMDE
FMCICS	FMPCF
FMCM1	FMRMH
FMCM2	FMRFH
FMDLH	FMRFH2
FMDH	FMSTR
FMEVNT	FMTM
FMIMS	FMXQH
FMIMVS	

- Nazwa formatu FMNONE to wartość specjalna, która wskazuje, że charakter danych w komunikacie nie jest zdefiniowany. W związku z tym menedżer kolejek nie próbuje konwersji, gdy komunikat jest pobierany z kolejki.

Uwaga: Jeśli parametr GMCONV został określony w wywołaniu MQGET dla komunikatu o nazwie formatu FMNONE, a zestaw znaków lub kodowanie komunikatu różni się od tego określonego w parametrze *MSGDSC*, to komunikat jest nadal zwracany w parametrze *BUFFER* (nie przyjmując innych błędów), ale wywołanie kończy się kodem zakończenia CCWARN i kodem przyczyny RC2110.

Parametr FMNONE może być używany, gdy rodzaj danych komunikatu oznacza, że nie wymaga konwersji, lub gdy aplikacje wysyłający i odbierający uzgodniły między sobą formularz, w którym dane komunikatu powinny zostać wysłane.

- Wszystkie inne nazwy formatu powodują, że komunikat jest przekazywany do wyjścia użytkownika w celu konwersji. Wyjście ma taką samą nazwę, jak format, poza dodatkami specyficznymi dla środowiska. Nazwy formatów podane przez użytkownika nie mogą rozpoczynać się od liter "MQ", ponieważ nazwy takie mogą być w konflikcie z nazwami formatów obsługiwanych w przyszłości.

Dane użytkownika w komunikacie mogą być konwertowane między dowolnymi obsługiwanymi zestawami znaków i kodowaniami. Należy jednak pamiętać, że jeśli komunikat zawiera co najmniej jedną strukturę nagłówka IBM MQ, nie można przekształcić komunikatu z zestawu znaków lub zestawu znaków zawierającego znaki dwubajtowe lub wielobajtowe dla dowolnych znaków, które są poprawne w nazwach kolejek. Kod przyczyny RC2111 lub RC2115 powoduje, że próba ta jest wykonywana, a komunikat jest zwracany bez konwersji. Zestaw znaków Unicode UCS-2 jest przykładem takiego zestawu znaków.

Po powrocie z wywołania MQGET następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:

- RCBRAK

Następujący kod przyczyny wskazuje, że komunikat mógł zostać pomyślnie przekształcony. Aplikacja musi sprawdzić pola *MDCSI* i *MDENC* w parametrze *MSGDSC*, aby dowiedzieć się, jakie są:

- RC2079

Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Uwaga: Interpretacja kodu przyczyny opisana w tym przykładzie jest prawdziwa dla konwersji wykonywanych przez wyjścia napisane przez użytkownika *tylko*, jeśli wyjście jest zgodne z wytycznymi przetwarzania.

9. W przypadku wbudowanych formatów wymienionych wcześniej menedżer kolejek może wykonać *domyślną konwersję* łańcuchów znaków w komunikacie, gdy zostanie podana opcja GMCONV. Domyślna konwersja umożliwia menedżerowi kolejek korzystanie z domyślnego zestawu znaków określonego przez instalację, który przybliża rzeczywisty zestaw znaków podczas przekształcania danych łańcuchowych. W rezultacie wywołanie MQGET może zakończyć się pomyślnie kodem zakończenia CCOK, zamiast kończenia z kodem CCWARN i kodem przyczyny RC2111 lub RC2115.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji danych łańcuchowych jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Można tego uniknąć, używając w łańcuchu tylko znaków, które są wspólne zarówno dla rzeczywistego zestawu znaków, jak i domyślnego zestawu znaków.

Domyślna konwersja ma zastosowanie zarówno do danych komunikatu aplikacji, jak i do pól znakowych w strukturach MQMD i MQMDE:

- Domyślna konwersja danych komunikatu aplikacji jest wykonywana tylko wtedy, gdy *wszystkie* z następujących elementów są prawdziwe:
 - Aplikacja określa GMCONV.
 - Komunikat zawiera dane, które muszą zostać przekształcone z lub do zestawu znaków, który nie jest obsługiwany.
 - Domyślna konwersja została włączona podczas instalowania lub restartowania menedżera kolejek.
 - W razie potrzeby domyślna konwersja pól znakowych w strukturach MQMD i MQMDE, jeśli dla menedżera kolejek włączona jest konwersja domyślna. Konwersja jest wykonywana nawet wtedy, gdy opcja GMCONV nie jest określona przez aplikację w wywołaniu MQGET.
10. Parametr *BUFFER* przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch; ograniczenie to ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr musi zostać zadeklarowany jako struktura, albo jako pole w zbiorze fizycznym.

Zadeklarowanie parametru jako struktury powoduje zwiększenie maksymalnej długości do 9999 bajtów, natomiast zadeklarowanie parametru jako pola w zbiorze fizycznym zwiększa maksymalną długość możliwą do około 32 kB.

Parametry

Wywołanie MQGET ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje kolejkę, z której ma zostać pobrany komunikat. Wartość *HOBJ* została zwrócona przez poprzednie wywołanie MQOPEN. Kolejka musi być otwarta z jedną lub więcej spośród następujących opcji (szczegółowe informacje na ten temat zawiera sekcja [“MQOPEN-obiekt otwarty”](#) na stronie 1346):

- OOINPS
- OOINPX
- POINPQ
- OOBROW

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Jeśli wartość parametru *BUFLEN* jest mniejsza niż długość komunikatu, to program *MSGDSC* nadal jest wprowadzany przez menedżer kolejek, niezależnie od tego, czy parametr *GMATM* został określony w parametrze *GMO* (patrz pole *GMOPT* opisane w sekcji [“MQGMO-Opcje Get-message”](#) na stronie 1089).

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, zwrócony komunikat ma przedrostek MQMDE, który jest poprzedzony danymi komunikatu aplikacji, ale *tylko*, jeśli co najmniej jedno z pól w MQMDE ma wartość niedomyślną. Jeśli wszystkie pola w tabeli MQMDE mają wartości domyślne, pomijane jest MQMDE. Nazwa formatu FMMDE w polu *MDFMT* w deskrypcie MQMD wskazuje, że jest obecna tabela MQMDE.

GMO (MQGMO)-wejście/wyjście

Opcje, które sterują działaniem MQGET.

Szczegółowe informacje na ten temat zawiera sekcja [“MQGMO-Opcje Get-message”](#) na stronie 1089.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość w bajtach obszaru *BUFFER*.

Wartość zero może być określona dla komunikatów, które nie mają danych, lub jeśli komunikat ma zostać usunięty z kolejki, a usunięte dane (w tym przypadku należy określić parametr *GMATM*).

Uwaga: Długość najdłuższej wiadomości, którą można odczytać z kolejki, jest nadawana przez atrybut kolejki *MaxMsgLength*; patrz [“Atrybuty dla kolejek”](#) na stronie 1395.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-dane wyjściowe

Obszar, który ma zawierać dane komunikatu.

Bufor musi być wyrównany na granicy odpowiedniej do charakteru danych w komunikacie. 4-bajtowe wyrównanie musi być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli wartość *BUFLEN* jest mniejsza niż długość komunikatu, to jak najwięcej komunikatu jest przenoszonych do produktu *BUFFER*; Oznacza to, czy parametr *GMATM* jest określony w parametrze *GMO* (więcej informacji znajduje się w polu *GMOPT* opisanym w sekcji [“MQGMO-Opcje Get-message”](#) na stronie 1089).

Zestaw znaków i kodowanie danych w programie *BUFFER* są nadawane przez pola *MDCSI* i *MDENC* zwracane w parametrze *MSGDSC*. Jeśli wartości te różnią się od wartości wymaganych przez odbiorcę, odbiorca musi dokonać konwersji danych komunikatu aplikacji na wymagany zestaw

znaków i kodowanie. Opcji GMCONV można użyć z wyjściem napisanego przez użytkownika w celu przeprowadzenia konwersji danych komunikatu (szczegółowe informacje na temat tej opcji znajdują się w sekcji “MQGMO-Opcje Get-message” na stronie 1089).

Uwaga: Wszystkie pozostałe parametry wywołania MQGET znajdują się w zestawie znaków i kodowaniu lokalnego menedżera kolejek (nadawanego przez atrybut menedżera kolejek produktu *CodedCharSetId* i ENNAT).

Jeśli wywołanie nie powiedzie się, zawartość buforu może być nadal zmieniona.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Długość komunikatu.

Jest to długość (w bajtach) danych aplikacji w *komunikacie*. Jeśli ta długość komunikatu jest większa niż *BUFLLEN*, w parametrze *BUFFER* zostaną zwrócone tylko *BUFLLEN* bajty (to znaczy, że komunikat jest obcinany). Jeśli wartość jest równa zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Jeśli wartość parametru *BUFLLEN* jest mniejsza niż długość komunikatu, program *DATLEN* nadal jest wprowadzany przez menedżer kolejek, niezależnie od tego, czy parametr *GMATM* został podany w parametrze *GMO* (więcej informacji na ten temat zawiera opis pola *GMOPT* opisany w sekcji “MQGMO-Opcje Get-message” na stronie 1089). Dzięki temu aplikacja może określić wielkość buforu wymaganego do obsługi danych komunikatu, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Jeśli jednak zostanie podana opcja GMCONV, a przekształcone dane komunikatu są zbyt długie, aby zmieścić się w programie *BUFFER*, wartość zwrócona dla *DATLEN* jest następująca:

- Długość danych *bez konwersji* dla formatów zdefiniowanych przez menedżera kolejek.

W takim przypadku, jeśli charakter danych powoduje jej rozszerzenie podczas konwersji, aplikacja musi przydzielić bufor większy niż wartość zwrócona przez menedżer kolejek dla produktu *DATLEN*.

- Wartość zwracana przez wyjście konwersji danych dla formatów zdefiniowanych przez aplikację.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Następujące kody przyczyn to te, które menedżer kolejek może zwrócić dla parametru *REASON* . Jeśli aplikacja określa opcję GMCONV, a program zewnętrzny napisany przez użytkownika jest wywoływany w celu przekształcenia niektórych lub wszystkich danych komunikatu, to jest to wyjście, które decyduje o tym, jaka wartość jest zwracana dla parametru *REASON* . W wyniku tego możliwe są wartości inne niż wartości przedstawione w dalszej części tej sekcji.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2120

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

RC2190

(2190, X'88E') Konwertowany łańcuch jest zbyt duży dla pola.

RC2150

(2150, X'866 ') Łańcuch DBCS nie jest poprawny.

RC2110

(2110, X'83E') Format komunikatu nie jest poprawny.

RC2243

(2243, X'8C3') Segmenty komunikatów mają różne identyfikatory CCSID.

RC2244

(2244, X'8C4') Segmenty komunikatów mają różne kodowania.

RC2209

(2209, X'8A1') Brak zablokowanego komunikatu.

RC2119

(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

RC2272

(2272, X'8E0') Dane komunikatu zostały częściowo przekształcone.

RC2145

(2145, X'861 ') Parametr buforu źródłowego jest niepoprawny.

RC2111

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

RC2113

(2113, X'841 ') Zpakowane kodowanie dziesiętne w komunikacie nie zostało rozpoznane.

RC2114

(2114, X'842 ') Kodowanie zmiennopozycyjne w komunikacie nie zostało rozpoznane.

RC2112

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

RC2143

(2143, X'85F') Parametr długości źródła nie jest poprawny.

RC2146

(2146, X'862 ') Parametr bufora docelowego jest niepoprawny.

RC2115

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

RC2117

(2117, X'845 ') Zpakowane-kodowanie dziesiętne określone przez odbiornik nierozpoznany.

RC2118

(2118, X'846 ') Kodowanie zmiennopozycyjne określone przez odbiornik nie jest rozpoznawane.

RC2116

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

RC2079

(2079, X'81F') Zwrócona została obcięta wiadomość (przetwarzanie zostało zakończone).

RC2080

(2080, X'820 ') Zwrócona została obcięta wiadomość (przetwarzanie nie zostało zakończone).

Jeśli *CMPCOD* to *CCFAIL*:

RC2004

(2004, X'7D4') Parametr bufora nie jest poprawny.

RC2005

(2005, X'7D5') Parametr długości bufora nie jest poprawny.

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010

(2010, X'7DA') Parametr długości danych nie jest poprawny.

RC2016

(2016, X'7E0') Liczba pobrań jest zablokowana dla kolejki.

RC2186

(2186, X'88A') Struktura opcji Get-message nie jest poprawna.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2241

(2241, X'8C1') Grupa komunikatów nie została zakończona.

RC2242

(2242, X'8C2') Komunikat logiczny nie został zakończony.

RC2259

(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246

(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2247

(2247, X'8C7') Opcje zgodności nie są poprawne.

RC2026

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

RC2250

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

RC2033

(2033, X'7F1') Brak dostępnego komunikatu.

RC2034

(2034, X'7F2') Przeglądaj kursor nie umieszczony na komunikacie.

RC2036

(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037

(2037, X'7F5') Kolejka nie jest otwarta dla danych wejściowych.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2024

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

RC2072

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2255

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

RC2090

(2090, X'82A') Interwał oczekiwania w MQGMO nie jest poprawny.

RC2256

(2256, X'8D0') Podano niewłaściwą wersję produktu MQGMO.

RC2257

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                      BUFLen : BUFFER : DATLEN :
C                      CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..
DMQGET      PR          EXTPROC('MQGET')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQGET
D GMO          112A
D* Length in bytes of the Buffer area
D BUFLen          10I 0 VALUE
D* Area to contain the message data
D BUFFER          *   VALUE
D* Length of the message
D DATLEN          10I 0
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0
```

MQINQ-zapytanie o atrybuty obiektu

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Poprawne są następujące typy obiektów:

- Kolejka
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- [“Składnia” na stronie 1329](#)
- [“Użycie notatek” na stronie 1329](#)
- [“Parametry” na stronie 1330](#)
- [“Deklaracja RPG” na stronie 1337](#)

Składnia

`MQINQ (HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)`

Użycie notatek

1. Zwracane wartości są obrazem stanu wybranych atrybutów. Nie ma gwarancji, że atrybuty nie zostaną zmienione, zanim aplikacja będzie mogła działać w oparciu o zwrócone wartości.
2. Po otwarciu kolejki modelowej tworzona jest dynamiczna kolejka lokalna. Jest to prawda, nawet jeśli kolejka modelowa zostanie otwarta w celu uzyskania informacji o jej atrybutach.

Atrybuty kolejki dynamicznej (z pewnymi wyjątkami) są takie same, jak atrybuty kolejki modelowej w momencie tworzenia kolejki dynamicznej. Jeśli następnie zostanie użyta wywołanie `MQINQ` w tej kolejce, menedżer kolejek zwróci atrybuty kolejki dynamicznej, a nie atrybuty kolejki modelowej. Patrz [Tabela 1](#), aby określić, które atrybuty kolejki modelowej są dziedziczone przez kolejkę dynamiczną.

3. Jeśli sprawdzany obiekt jest kolejką aliasową, to wartości atrybutów zwracane przez wywołanie `MQINQ` są wartościami kolejki aliasowej, a nie wartościami kolejki podstawowej, do której alias jest tłumaczący.
4. Jeśli sprawdzany obiekt jest kolejką klastra, atrybuty, które mogą być zapytania, zależą od sposobu otwierania kolejki:
 - Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądanych lub ustawionych, musi istnieć lokalna instancja kolejki klastra, aby możliwe było pomyślne wykonanie tej kolejki. W tym przypadku atrybuty, które mogą być dociekliwe, są poprawne dla kolejek lokalnych.
 - Jeśli kolejka klastra jest otwarta tylko do zapytania lub zapytania i wyjścia, można uzyskać dostęp tylko do następujących atrybutów: atrybut *QType* ma w tym przypadku wartość `QTCLUS`:
 - CAQD
 - CAQN
 - IADBND
 - IADPER
 - IADPRI
 - IAIPUT
 - IAQTYP

Jeśli kolejka klastra jest otwarta bez ustalonego powiązania (to jest `OOBNDN` określone w wywołaniu `MQOPEN` lub `OQBNDQ` określone, gdy atrybut *DefBind* ma wartość `BNDNOT`), kolejne wywołania `MQINQ` dla kolejki mogą zapytać o różne instancje kolejki klastra, chociaż zwykle wszystkie instancje mają takie same wartości atrybutów.

Więcej informacji na temat kolejek klastra zawiera sekcja [Konfigurowanie klastra menedżera kolejek](#).

5. Jeśli wymagane jest zapytanie o pewną liczbę atrybutów, a następnie niektóre z nich mają być ustawione za pomocą wywołania MQSET, można wygodnie ustawić pozycję na początku tablic selektorów atrybutu, które mają zostać ustawione, tak aby dla tabeli MQSET można było użyć tych samych macierzy (z licznymi zmniejszonymi liczebnościami).
6. Jeśli wystąpi więcej niż jedna z sytuacji ostrzegawczych (patrz parametr *CMPCOD*), zwrócony kod przyczyny ma wartość *pierwsza* na następującej liście, która ma zastosowanie:
 - a. RC2068
 - b. RC2022
 - c. RC2008
7. Więcej informacji na temat atrybutów obiektów zawiera sekcja:
 - [“Atrybuty dla kolejek” na stronie 1395](#)
 - [“Atrybuty dla list nazw” na stronie 1424](#)
 - [“Atrybuty definicji procesów” na stronie 1425](#)
 - [“Atrybuty dla menedżera kolejek” na stronie 1427](#)
8. Nowa kolejka lokalna SYSTEM.ADMIN.COMMAND.EVENT jest używany do kolejowania komunikatów generowanych za każdym razem, gdy wydawane są komendy. Komunikaty są umieszczane w tej kolejce dla większości komend, w zależności od tego, w jaki sposób ustawiony jest atrybut menedżera kolejek CMDEV:
 - Komunikaty zdarzeń komendy ENABLED są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend.
 - Komunikaty zdarzeń komendy NODISPLAY są generowane i umieszczane w kolejce dla wszystkich komend zakończonych powodzeniem, innych niż komenda DISPLAY (MQSC), oraz komenda Inquire (PCF).
 - Komunikaty zdarzenia DISABLED-zdarzenia komendy nie są generowane (jest to początkowa wartość domyślna menedżera kolejek).

Parametry

Wywołanie MQINQ ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt (dowolnego typu) z wymaganymi atrybutami. Uchwyt musi zostać zwrócony przez poprzednie wywołanie MQOPEN, które określiło opcję OOINQ.

SELCNT (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Liczba selektorów.

Jest to liczba selektorów, które są dostarczane w macierzy *SELS*. Jest to liczba atrybutów, które mają zostać zwrócone. Wartość zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

SELS (10-cyfrowa liczba całkowita ze znakiem x SELCNT)-dane wejściowe

Tablica selektorów atrybutów.

Jest to tablica selektorów atrybutów *SELCNT*; każdy selektor identyfikuje atrybut (liczba całkowita lub znak) z wymaganą wartością.

Każdy selektor musi być poprawny dla typu obiektu reprezentowanego przez produkt *HOB*. W przeciwnym razie wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2067.

W specjalnym przypadku kolejek:

- Jeśli selektor nie jest poprawny dla kolejek o typie *any*, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2067.
- Jeśli selektor ma zastosowanie *tylko* do kolejek typu lub typów innych niż typ obiektu, wywołanie powiedzie się z kodem zakończenia CCWARN i kodem przyczyny RC2068.
- Jeśli zapytanie jest kolejką klastra, poprawne selektory zależą od sposobu, w jaki została rozstrzygnięta kolejka. Więcej informacji na ten temat zawiera uwaga dotycząca użycia 4.

Selektory mogą być określane w dowolnej kolejności. Wartości atrybutów, które odpowiadają selektorom atrybutów całkowitych (IA* selektory), są zwracane w produkcie *INTATR* w tej samej kolejności, w jakiej te selektory występują w produkcie *SELS*. Wartości atrybutów, które odpowiadają selektorom atrybutów znakowych (CA* selektory), są zwracane w produkcie *CHRATR* w tej samej kolejności, w jakiej występują te selektory. Selektory IA* można przeplatać się z selektorami CA*; ważne jest tylko to, że kolejność względna w każdym typie jest istotna.

Uwaga:

1. Selektory atrybutów całkowitych i atrybutów znakowych są przydzielane w dwóch różnych zakresach; selektory IA* znajdują się w zakresie IAFRST przez IALAST, a selektory CA* w zakresie CAFRST poprzez CALAST.

Dla każdego zakresu wartości stałe IALSTU i CALSTU definiują najwyższą wartość akceptowania przez menedżer kolejek.

2. Jeśli wszystkie selektory IA* występują jako pierwsze, te same numery elementów mogą być używane do adresowania odpowiednich elementów w macierzach *SELS* i *INTATR*.

Atrybuty, które można uzyskać do zapytania, są wymienione w poniższych tabelach. W przypadku selektorów CA* stała definiująca długość łańcucha wynikowego w bajtach w programie *CHRATR* jest podana w nawiasach.

Tabela 171. Selektory atrybutów <i>MQINQ</i> dla kolejek.		
Wyjaśnienie uwag znajduje się w dolnej części tabeli.		
Selektor	Opis	Uwaga
CAALTD	Data ostatniej zmiany (LNDATE).	1
CAALTT	Czas ostatniej zmiany (LNTIME).	1
CABRQN	Nadmierna liczba wycofanych komunikatów (LNQN).	5
CABASQ	Nazwa kolejki, do której alias jest tłumaczona (LNQN).	
CACFSN	Nazwa struktury narzędzia CF (Coupling-facility structure name-LNCFSN).	3
CACLN	Nazwa klastra (LNCLUN).	1
CACLNL	Lista nazw klastrów (LNNLN).	1
CACRTD	Data utworzenia kolejki (LNCRTD).	

Tabela 171. Selektory atrybutów MQINQ dla kolejek.

Wyjaśnienie uwag znajduje się w dolnej części tabeli.

(kontynuacja)

Selektor	Opis	Uwaga
CACRTT	Czas utworzenia kolejki (LNCRTT).	
CAINIQ	Nazwa kolejki inicjuj. (LNQN).	
CAPRON	Nazwa definicji procesu (LNPRON).	
CAQD	Opis kolejki (LNQD).	
CAQN	Nazwa kolejki (LNQN).	
CARQMN	Nazwa zdalnego menedżera kolejek (LNQMN).	
CARQN	Nazwa kolejki zdalnej, która jest znana w zdalnym menedżerze kolejek (LNQN).	
CATRGD	Dane wyzwalacza (LNTRGD).	5
CAXQN	Nazwa kolejki transmisji (LNQN).	
IABTHR	Próg wycofania.	5
IACDEP	Liczba komunikatów w kolejce.	
IADBND	Powiązanie domyślne.	1
IADINP	Domyślna opcja open-for-input.	5
IADPER	Domyślna trwałość komunikatu.	
IADPRI	Domyślny priorytet komunikatu.	5
IADEFT	Typ definicji kolejki.	
IADIST	Obsługa listy dystrybucyjnej.	2
IAHGB	Określa, czy licznik wycofań ma być twardniejący.	5
IAIGET	Określa, czy operacje pobierania są dozwolone.	
IAIPUT	Określa, czy operacje put są dozwolone.	
IAMLEN	Maksymalna długość komunikatu.	
IAMDEP	Maksymalna liczba komunikatów dozwolonych w kolejce.	
IAMDS	Określa, czy priorytet komunikatu jest istotny.	5
IAOIC	Liczba wywołań MQOPEN, które mają otwartą kolejkę dla danych wejściowych.	
IAOOC	Liczba wywołań MQOPEN, które mają otwartą kolejkę dla danych wyjściowych.	
IAQDHE	Atrybut elementu sterującego dla zdarzeń wysokiego zapętnienia kolejki.	4, 5
IAQDHL	Górny limit głębokości kolejki.	4, 5
IAQDLE	Atrybut elementu sterującego dla zdarzeń o niskiej głębokości kolejki.	4, 5
IAQDLL	Niski limit głębokości kolejki.	4, 5
IAQDME	Atrybut elementu sterującego dla zdarzeń maksymalnej głębokości kolejki.	4, 5
IAQSI	Limit czasu dla usługi kolejki.	4, 5
IAQSIE	Atrybut elementu sterującego dla zdarzeń odstępu czasu usługi kolejki.	4, 5

Tabela 171. Selektory atrybutów MQINQ dla kolejek.

Wyjaśnienie uwag znajduje się w dolnej części tabeli.

(kontynuacja)

Selektor	Opis	Uwaga
IAQTYP	Typ kolejki.	
IAQSGD	Dyspozycja grupy współużytkowania kolejki.	3
IARINT	Interwał czasu przechowywania kolejki.	5
IASCOP	Zasięg definicji kolejki.	4, 5
IASHAR	Określa, czy kolejka może być współużytkowana dla danych wejściowych.	
IATRGC	Sterowanie wyzwalaczem.	
IATRGD	Wyzwalacz uruchamiany zapełnieniem.	5
IATRGP	Priorytet komunikatu progowego dla wyzwalaczy.	5
IATRGT	Typ wyzwalacza.	
IAUSAG	Użycie.	
CLWLUSEQ	Użyj kolejek zdalnych.	

Uwaga:

1. Obsługiwane w systemach AIX, HP-UX, z/OS, IBM i, Solaris, Windows oraz IBM MQ MQI clients połączonych z tymi systemami.
2. Obsługiwane w klientach AIX, HP-UX, IBM i, Solaris, Windows oraz IBM MQ połączonych z tymi systemami.
3. Obsługiwane w systemie z/OS.
4. Nieobsługiwane w produkcie z/OS.
5. Nieobsługiwane w systemie VSE/ESA.

Tabela 172. Selektory atrybutów MQINQ dla list nazw.

Wyjaśnienie uwag można znaleźć w sekcji [Uwagi](#).

Selektor	Opis	Uwaga
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CALSTD	Opis listy nazw (LNNLD)	1
CALSTN	Nazwa obiektu listy nazw (LNNLN)	1
CANAMS	Nazwy na liście nazw (LNQN x Liczba nazw na liście)	1
IANAMC	Liczba nazw na liście nazw	1
IAQSGD	Dyspozycja grupy współużytkowania kolejki	3

Tabela 173. Selektory atrybutów MQINQ dla definicji procesów.

Wyjaśnienie uwag można znaleźć w sekcji [Uwagi](#) .

Selektor	Opis	Uwaga
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CAAPPI	Identyfikator aplikacji (LNPROA)	5
CAENVD	Dane środowiska (LNPROE)	5
CAPROD	Opis definicji procesu (LNPROD)	5
CAPRON	Nazwa definicji procesu (LNPRON)	5
CAUSRD	Dane użytkownika (LNPROU)	5
IAAPPT	Typ aplikacji	5
IAQSGD	Dyspozycja grupy współużytkowania kolejki	3

Tabela 174. Selektory atrybutów MQINQ dla menedżera kolejek.

Wyjaśnienie uwag można znaleźć w sekcji [Uwagi](#) .

Selektor	Opis	Uwaga
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CACADX	Nazwa wyjścia automatycznej definicji kanału (LNEXN)	1
CACLWD	Dane przekazane do wyjścia obciążenia klastra (LNEXDA)	1
CACLWX	Nazwa wyjścia obciążenia klastra (LNEXN)	1
CACMDQ	Nazwa kolejki wejściowej komend systemowych (LNQN)	5
CADLQ	Nazwa kolejki niedostarczonych komunikatów (LNQN)	5
CADXQN	Domyślna nazwa kolejki transmisji (LNQN)	5
CAQMD	Opis menedżera kolejek (LNQMD)	5
Identyfikator CAQMID	Identyfikator menedżera kolejek (LNQMID)	1
CAQMN	Nazwa lokalnego menedżera kolejek (LNQMN)	5
CAQSGN	Nazwa grupy współużytkowania kolejki (LNQSGN)	3
KARTA	Nazwa klastra, dla którego menedżer kolejek udostępnia usługi repozytorium (LNQMN)	1
CARPNL	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których menedżer kolejek udostępnia usługi repozytorium (LNNLN)	1
CMDEV	Atrybut sterujący, który określa, czy komunikaty generowane przy wydawanych komendach są umieszczane w kolejce	8
IAAUTE	Atrybut elementu sterującego dla zdarzeń uprawnień	4, 5
IACAD	Atrybut elementu sterującego dla definicji kanału automatycznego	2
IACADE	Atrybut elementu sterującego dla zdarzeń automatycznej definicji kanału	2

Tabela 174. Selektory atrybutów MQINQ dla menedżera kolejek.

Wyjaśnienie uwag można znaleźć w sekcji [Uwagi](#) .

(kontynuacja)

Selektor	Opis	Uwaga
IACLXQ	Domyślny typ kolejki transmisji klastra	4
IACLWL	Długość obciążenia klastra	1
IACCSI	Identyfikator kodowanego zestawu znaków	5
IACMDL	Poziom komend obsługiwany przez menedżer kolejek	5
IACFGE	Atrybut elementu sterującego dla zdarzeń konfiguracji	3
IADIST	Obsługa listy dystrybucyjnej	2
IAINHE	Atrybut elementu sterującego dla zdarzeń zablokowanej	4, 5
IACLE	Atrybut elementu sterującego dla zdarzeń lokalnych	4, 5
IAMHND	Maksymalna liczba uchwytów	5
IAMLEN	Maksymalna długość komunikatu	5
IAMPRI	Maksymalny priorytet	5
IAMUNC	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy	5
IAPFME	Atrybut elementu sterującego dla zdarzeń wydajności	4, 5
IAPLAT	Platforma, na której znajduje się menedżer kolejek	5
IARMTE	Atrybut elementu sterującego dla zdarzeń zdalnych	4, 5
IASSE	Atrybut elementu sterującego uruchamiania zdarzeń zatrzymania	4, 5
IASYNC	Dostępność punktu synchronizacji	5
IATRLFT	Czas życia nieużywanych tematów nieadministracyjnych	
IATRGI	Interwał wyzwalacza	5

IACNT (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Liczba atrybutów całkowitych.

Jest to liczba elementów w tablicy *INTATR* . Wartość zero jest poprawną wartością.

Jeśli jest to co najmniej liczba selektorów IA* w parametrze *SELS* , zwracane są wszystkie żądane atrybuty całkowitoliczbowe.

INTATR (10-cyfrowa liczba całkowita ze znakiem x IACNT)-dane wyjściowe

Tablica atrybutów całkowitoliczbowych.

Jest to tablica wartości atrybutu całkowitoliczbowego *IACNT* .

Wartości atrybutów całkowitych są zwracane w tej samej kolejności, w jakiej znajdują się selektory IA* w parametrze *SELS* . Jeśli tablica zawiera więcej elementów niż liczba selektorów IA*, nadmiarowe elementy są niezmienione.

Jeśli *HOB*J reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, określona wartość *IAVNA* jest zwracana dla odpowiadającego mu elementu w tablicy *INTATR* .

CALEN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Długość buforu atrybutów znakowych.

Jest to długość w bajtach parametru *CHRATR*.

Musi to być co najmniej suma długości żądanych atrybutów znakowych (patrz *SELS*). Wartość zero jest poprawną wartością.

CHRATR (1 bajt łańcucha znaków x CALEN)-dane wyjściowe

Atrybuty znaków.

Jest to bufor, w którym zwracane są atrybuty znakowe, konkatelowane razem. Długość buforu jest nadawana przez parametr *CALEN*.

Atrybuty znaków są zwracane w tej samej kolejności, w jakiej znajdują się selektory CA* w parametrze *SELS*. Długość każdego łańcucha atrybutu jest stała dla każdego atrybutu (patrz *SELS*), a wartość w niej jest dopełniona do prawej strony, jeśli jest to konieczne, z odstępami. Jeśli bufor jest większy niż wymagany, aby zawierał wszystkie żądane atrybuty znaków (w tym dopełnianie), liczba bajtów poza ostatnią zwróci wartość atrybutu nie zmienia się.

Jeśli *HOBJ* reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, łańcuch znaków składający się w całości z gwiazdek (*) jest zwracany jako wartość tego atrybutu w produkcie *CHRATR*.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2008

(2008, X'7D8') Niewystarczająca ilość miejsca na atrybuty znaków.

RC2022

(2022, X'7E6') Niewystarczająca ilość miejsca na atrybuty całkowite.

RC2068

(2068, X'814 ') Selektor nie ma zastosowania do typu kolejki.

Jeśli *CMPCOD* to CCFail:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2006

(2006, X'7D6') Długość atrybutów znakowych nie jest poprawna.

RC2007

(2007, X'7D7') Łańcuch atrybutów znakowych nie jest poprawny.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2021

(2021, X'7E5') Liczba atrybutów całkowitych nie jest poprawna.

RC2023

(2023, X'7E7') Tablica atrybutów Integer nie jest poprawna.

RC2038

(2038, X'7F6') Kolejka nie jest otwarta dla zapytania.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2065

(2065, X'811 ') Count of selectors not valid.

RC2067

(2067, X'813 ') Selektor atrybutu nie jest poprawny.

RC2066

(2066, X'812 ') Count of selectors too large.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQINQ(HCONN : HOBJ : SELCNT :
C                      SELS(1) : IACNT : INTATR(1) :
C                      CALEN : CHRATR : CMPCOD :
C                      REASON)

```

Definicja prototypu dla wywołania to:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQINQ          PR          EXTPROC('MQINQ')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Count of selectors
D SELCNT          10I 0 VALUE
D* Array of attribute selectors
D SELS          10I 0

```

```

D* Count of integer attributes
D IACNT 10I 0 VALUE
D* Array of integer attributes
D INTATR 10I 0
D* Length of character attributes buffer
D CALEN 10I 0 VALUE
D* Character attributes
D CHRATR * VALUE
D* Completion code
D CMPCOD 10I 0
D* Reason code qualifying CMPCOD
D REASON 10I 0

```

MQINQMP-właściwość komunikatu Inquire

Wywołanie MQINQMP zwraca wartość właściwości komunikatu.

- [“Składnia” na stronie 1338](#)
- [“Parametry” na stronie 1338](#)
- [“Deklaracja RPG” na stronie 1342](#)

Składnia

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parametry

Wywołanie MQINQMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwytami połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *Hmsg*.

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy HCUNAS, należy ustawić poprawne połączenie w wątku, w którym można uzyskać dostęp do właściwości uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Jest to uchwyt komunikatu, który ma zostać wyświetlony. Wartość została zwrócona przez poprzednie wywołanie **MQCRTMH**.

INQOPT (MQIMPO)-dane wejściowe

Szczegółowe informacje zawiera opis typu danych [MQIMPO](#).

PRNAME (MQCHARV)-dane wejściowe

W tym temacie opisano nazwę właściwości, która ma zostać zapytana.

Jeśli nie można znaleźć żadnej właściwości o tej nazwie, wywołanie nie powiedzie się z powodu RC2471.

Na końcu nazwy właściwości można użyć znaku procentu (%). Znak wieloznaczny zastępuje zero lub więcej znaków, w tym znak kropki (.). Dzięki temu aplikacja może uzyskać dostęp do wartości wielu właściwości. Wywołaj komendę MQINQMP z opcją IPINQF, aby pobrać pierwszą zgodną właściwość i ponownie z opcją IPINQN, aby uzyskać następną pasującą właściwość. Jeśli nie są dostępne żadne dodatkowe właściwości, wywołanie kończy się niepowodzeniem z kodem RC2471. Jeśli pole *ReturnedName* struktury *InqPropzostanie* zainicjowane z adresem lub przesuniętą dla zwróconej nazwy właściwości, zostanie ona zakończona po powrocie z tabeli MQINQMP z nazwą właściwości, która została dopasowana. Jeśli pole *VSBufSize* w strukturze *ReturnedName* w strukturze *InqPropOpts* jest mniejsze niż długość zwróconej nazwy właściwości, to kod zakończenia jest ustawiony na wartość CCFAIL z przyczyną RC2465.

Właściwości, które mają znane synonimy, są zwracane w następujący sposób:

1. Właściwości z przedrostkiem "mqps." są zwracane wraz z nazwą właściwości IBM MQ . Na przykład "MQTopicString" jest nazwą zwracaną, a nie "mqps.Top".
2. Właściwości z przedrostkiem "jms." lub "mcd." są zwracane jako nazwa pola nagłówka JMS . Na przykład "JMSExpiration" jest nazwą zwracaną, a nie "jms.Exp".
3. Właściwości z przedrostkiem "usr." są zwracane bez tego przedrostka. Na przykład zwracana jest wartość "Color", a nie "usr.Color".

Właściwości z synonimami są zwracane tylko jeden raz.

W języku programowania RPG następujące zmienne makra są zdefiniowane dla zapytania o wszystkie właściwości i wszystkie właściwości, które rozpoczynają się od "usr.":

INQALL

Sprawdź, czy wszystkie właściwości komunikatu są dostępne.

INQUSR

Sprawdź wszystkie właściwości komunikatu, które zaczynają się od "usr.". Zwrócona nazwa jest zwracana bez użycia "usr." przedrostek.

Jeśli określono IPINQN, ale nazwa została zmieniona od czasu poprzedniego wywołania lub jest to pierwsze wywołanie, to IPINQF jest dorozumiana.

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja Nazwy właściwości i Ograniczenia dotyczące nazw właściwości .

PRPDSC (MQPD)-dane wyjściowe

Ta struktura jest używana do definiowania atrybutów właściwości, w tym elementów, które są wykonywane, jeśli właściwość nie jest obsługiwana, kontekst komunikatu, do którego należy właściwość oraz jakie komunikaty należy skopiować do tej właściwości. Szczegółowe informacje na temat tej struktury zawiera sekcja MQPD .

TYPE (10-cyfrowa liczba całkowita ze znakiem)-input/output

W przypadku powrotu z wywołania MQINQMP ten parametr jest ustawiony na typ danych *Wartość*. Typ danych może mieć jedną z następujących wartości:

TYPBOL

Wartość boolowska.

TYPBST

łańcuch bajtowy.

TYPI8

8-bitowa liczba całkowita ze znakiem.

TYPI16

16-bitowa liczba całkowita ze znakiem.

TYPI32

32-bitowa liczba całkowita ze znakiem.

TYPI64

64-bitowa liczba całkowita ze znakiem.

TYPF32

32-bitowa liczba zmiennoprzecinkowa.

TYPF64

64-bitową liczbę zmiennopozycyjną.

TYPSTR

łańcuch znaków.

TYPNUL

Właściwość istnieje, ale ma wartość NULL.

Jeśli typ danych wartości właściwości nie jest rozpoznawany, zwracana jest wartość TYPSTR, a reprezentacja łańcuchowa wartości jest umieszczana w obszarze *Wartość* . Reprezentację łańcuchową typu danych można znaleźć w polu *IPTYP* parametru *IPOPT* . Kod zakończenia ostrzeżenia jest zwracany z przyczyną RC2467.

Dodatkowo, jeśli zostanie podana opcja IPCTYP, zażądano konwersji wartości właściwości. Użyj opcji *Type* (Typ) jako danych wejściowych, aby określić typ danych, który ma być zwracany jako właściwość. Szczegółowe informacje na temat konwersji typów danych można znaleźć w opisie opcji IPCTYP [“MQIMPO-Zapytanie o opcje właściwości komunikatu” na stronie 1117](#) .

Jeśli nie zostanie wysłane żądanie konwersji typów, można użyć następującej wartości na wejściu:

TYPAST

Wartość właściwości jest zwracana bez przekształcania jej typu danych.

VALLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość w bajtach obszaru *Wartość* .

Podaj wartość zero dla właściwości, dla których nie jest wymagana zwracana wartość. Mogą to być właściwości, które zostały zaprojektowane przez aplikację w celu posiadania wartości NULL lub pustego łańcucha. Należy również określić wartość zero, jeśli została określona opcja IPQLEN. W tym przypadku nie jest zwracana żadna wartość.

VALUE (1-bajtowy łańcuch bitowy x VALLEN)-dane wyjściowe

Jest to obszar, który ma zawierać wartość właściwości inquired. Bufor powinien być wyrównany do granicy odpowiedniej dla zwracanej wartości. Niepowodzenie może spowodować wystąpienie błędu, gdy wartość zostanie później uzyskana.

Jeśli wartość *VALLEN* jest mniejsza niż długość wartości właściwości, to jak większość wartości właściwości jest przenoszona do wartości *VALUE* , a wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i z przyczyną RC2469.

Zestaw znaków danych w polu *VALUE* jest nadawany przez pole IPRETCSI w parametrze INQOPT. Kodowanie danych w polu *VALUE* jest podawane przez pole IPRETENC w parametrze INQOPT.

Jeśli parametr *VALLEN* ma wartość zero, wartość *VALUE* nie jest przywołana.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Jest to długość w bajtach rzeczywistej wartości właściwości, która została zwrócona w obszarze *Wartość* .

Jeśli wartość *DataLength* jest mniejsza niż długość wartości właściwości, wartość *DataLength* jest nadal wprowadzana w odpowiedzi z wywołania MQINQMP. Dzięki temu aplikacja może określić wielkość buforu wymaganego do uwzględnienia wartości właściwości, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Mogą zostać również zwrócone następujące wartości.

Jeśli parametr *Type* jest ustawiony na wartość TYPSTR lub TYPBST:

VLEMP

Właściwość istnieje, ale nie zawiera żadnych znaków ani bajtów.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to CCWARN:

RC2492

(2492, X'09BC') Zwrócona nazwa właściwości nie została przekształcona.

RC2466

(2466, X'09A2') Wartość właściwości nie została przekształcona.

RC2467

(2467, X'09A3') Typ danych właściwości nie jest obsługiwany.

RC2421

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CMPCOD* to CCFAIL:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'086D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2004

(2004, X'07D4') Parametr Wartość nie jest poprawny.

RC2005

(2005, X'07D5') Parametr długości wartości nie jest poprawny.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010

(2010, X'07DA') Parametr długości danych nie jest poprawny.

RC2464

(2464, X'09A0') Zapytanie o strukturę opcji właściwości komunikatu nie jest poprawne.

RC2460

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2064

(2046, X'07F8') Opcje nie są poprawne lub niespójne.

RC2482

(2482, X'09B2') Struktura deskryptora właściwości nie jest poprawna.

RC2470

(2470, X'09A6') Konwersja z rzeczywistego na żądany typ danych nie jest obsługiwana.

RC2442

(2442, X'098A') Niepoprawna nazwa właściwości.

RC2465

(2465, X'09A1') Nazwa właściwości jest zbyt duża dla zwróconego buforu nazw.

RC2471

(2471, X'09A7') Właściwość nie jest dostępna.

RC2469

(2469, X'09A5') Wartość właściwości jest zbyt duża dla obszaru Wartość.

RC2472

(2472, X'09A8') Napotkano błąd formatu liczb w danych wartości.

RC2473

(2473, X'09A9') Niepoprawny żądany typ właściwości.

RC2111

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

RC2071

(2071, X'0871 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów można znaleźć w:

- [IBM MQ for z/OS : komunikaty, zakończenie i kody przyczyny dla IBM MQ for z/OS](#)
- [Komunikaty i kody przyczyny dla wszystkich pozostałych platform IBM MQ](#)

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQINQMP(HCONN : HMSG : INQOPT :
                                           PRNAME : PRPDSC : TYPE :
                                           VALLEN : VALUE : DATLEN :
                                           CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

DMQINQMP      PR      EXTPROC('MQINQMP')
D* Connection handle
D HCONN      10I 0 VALUE
D* Message handle
D HMSG      20I 0 VALUE
D* Options that control the action of MQINQMP
D INQOPT      72A
D* Property name
D PRNAME      32A
D* Property descriptor
D PRPDSC      24A
D* Property data type
D TYPE      10I 0
D* Length in bytes of the Value area
D VALLEN      10I 0 VALUE
D* Property value
D VALUE      * VALUE
D* Length of the property value
D DATLEN      10I 0
D* Completion code
D CMPCOD      10I 0
D* Reason code qualifying CompCode
D REASON      10I 0

```

MQMHBUF-Przekształć uchwyt komunikatu w bufor

MQMHBUF przekształca uchwyt komunikatu w bufor i jest odwrotnym wywołaniem wywołania MQBUFMH.

- [“Składnia” na stronie 1343](#)
- [“Użycie notatek” na stronie 1343](#)
- [“Parametry” na stronie 1343](#)

- [“Deklaracja RPG” na stronie 1345](#)

Składnia

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Użycie notatek

MQMHBUF przekształca uchwyt komunikatu w bufor.

Można go używać z wyjściem interfejsu API MQGET w celu uzyskania dostępu do określonych właściwości, za pomocą funkcji API właściwości komunikatu, a następnie przekazać te właściwości w buforze z powrotem do aplikacji zaprojektowanej w celu użycia nagłówków MQRFH2, a nie uchwytów komunikatów.

To wywołanie jest odwrotnym wywołaniem wywołania MQBUFMH, którego można użyć do analizowania właściwości komunikatu z buforu do uchwytu komunikatu.

Parametry

Wywołanie MQMHBUF ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość *HCONN* musi być zgodna z uchwytym połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *HMSG*.

Jeśli uchwyt komunikatu został utworzony za pomocą komendy HCUNAS, należy ustawić poprawne połączenie w wątku usuwania uchwytu komunikatu. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie zakończy się niepowodzeniem z kodem RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt jest uchwytym komunikatu, dla którego wymagany jest bufor.

Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

MHBOPT (MQMHBO)-dane wejściowe

Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem, w jaki bufor jest generowany z uchwytów komunikatów.

Szczegółowe informacje na ten temat zawiera sekcja [“MQBMHO-Opcje uchwytu buforu do obsługi komunikatów” na stronie 1030](#).

PRNAME (MQCHARV)-dane wejściowe

Nazwa właściwości lub właściwości, które mają zostać umieszczone w buforze.

Jeśli nie można znaleźć żadnej właściwości zgodnej z nazwą, wywołanie kończy się niepowodzeniem z kodem RC2471.

Znaki wieloznaczne

Aby umieścić w buforze więcej niż jedną właściwość, można użyć znaku wieloznacznego. Aby to zrobić, należy użyć znaku procentu (%) na końcu nazwy właściwości. Ten znak wieloznaczny zastępuje zero lub więcej znaków, w tym znak kropki (.).

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja [Nazwy właściwości i Ograniczenia dotyczące nazw właściwości](#).

MSGDSC (MQMD)-wejście/wyjście

Struktura *MSGDSC* opisuje zawartość obszaru buforu.

W przypadku danych wyjściowych pola *Encoding*, *CodedCharSetId* i *Format* są ustawione tak, aby poprawnie opisywać kodowanie, identyfikator zestawu znaków i format danych w obszarze buforu, jak zostało to zapisane w wywołaniu.

Dane w tej strukturze znajdują się w zestawie znaków i kodowaniu aplikacji.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

BUFLEN to długość obszaru buforu (w bajtach).

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście/wyjście

BUFFER definiuje obszar zawierający bufor komunikatów. W przypadku większości danych, należy wyrównać bufor na granicy 4-bajtowej.

Jeśli *BUFFER* zawiera dane znakowe lub numeryczne, ustaw wartości pól *CodedCharSetId* i *Encoding* w parametrze *MSGDSC* na wartości odpowiednie dla danych. Umożliwia to przekształcenie danych, jeśli to konieczne.

Jeśli właściwości znajdują się w buforze komunikatów, są one opcjonalnie usuwane, a później stają się dostępne z uchwytu komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że adres dowolnego typu danych może być określony jako parametr.

If the *BUFLEN* parameter is zero, *BUFFER* is not referred to. W tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

DATLEN to długość (w bajtach) zwracanych właściwości w buforze. Jeśli wartość wynosi zero, żadne właściwości nie są zgodne z wartością podaną w *PRNAME*, a wywołanie kończy się niepowodzeniem z kodem przyczyny RC2471.

Jeśli wartość *BUFLEN* jest mniejsza niż długość wymagana do zapisania właściwości w buforze, wywołanie *MQMHBUF* kończy się niepowodzeniem z kodem RC2469, ale wartość ta jest nadal wprowadzana do produktu *DATLEN*. Dzięki temu aplikacja może określić wielkość buforu wymaganego do dostosowania właściwości, a następnie ponownie wywołać wywołanie z wymaganym *BUFLEN*.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2501

(2501, X'095C') Uchwyt komunikatu do struktury opcji buforu nie jest poprawny.

RC2004

(2004, X'07D4') Parametr buforu nie jest poprawny.

RC2005

(2005, X'07D5') Parametr długości buforu nie jest poprawny.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009

(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010

(2010, X'07DA') Parametr długości danych nie jest poprawny.

RC2460

(2460, X'099C') Uchwyt komunikatu nie jest poprawny.

RC2026

(2026, X'07EA') deskryptor komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2442

(2442, X'098A') Nazwa właściwości jest niepoprawna.

RC2471

(2471, X'09A7') Właściwość nie jest dostępna.

RC2469

(2469, X'09A5') Wartość parametru BufferLength jest zbyt mała, aby można było zawierać określone właściwości.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQMHBUFF(HCONN : HMSG : MHBOPT :
                                           PRNAME : MSGDSC : BUFLN :
                                           BUFFER : DATLEN :
                                           CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

DMQMHBUFF      PR              EXTPROC('MQMHBUFF')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              20I 0 VALUE
D* Options that control the action of MQMHBUFF
D MHBOPT              12A
D* Property name
D PRNAME              32A
D* Message descriptor
D MSGDSC              364A
D* Length in bytes of the Buffer area
D BUFLN              10I 0 VALUE
D* Area to contain the properties
D BUFFER              *   VALUE
D* Length of the properties
D DATLEN              10I 0

```

D* Completion code	
D CMPCOD	10I 0
D* Reason code qualifying CompCode	
D REASON	10I 0

MQOPEN-obiekt otwarty

Wywołanie MQOPEN ustanawia dostęp do obiektu.

Poprawne są następujące typy obiektów:

- Kolejka (w tym listy dystrybucyjne)
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat
- [“Składnia” na stronie 1346](#)
- [“Użycie notatek” na stronie 1346](#)
- [“Parametry” na stronie 1351](#)
- [“Deklaracja RPG” na stronie 1357](#)

Składnia

MQOPEN (*HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON*)

Użycie notatek

1. Otwarty obiekt jest jednym z następujących:

- Kolejka, w celu:
 - Pobieranie lub przeglądanie komunikatów (za pomocą wywołania MQGET)
 - Umieszczanie komunikatów (za pomocą wywołania MQPUT)
 - Sprawdź atrybuty kolejki (za pomocą wywołania MQINQ)
 - Ustaw atrybuty kolejki (za pomocą wywołania MQSET)

Jeśli kolejka nazwana jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna.

Lista dystrybucyjna jest specjalnym typem obiektu kolejki, który zawiera listę kolejek. Można je otwierać w celu umieszczania komunikatów, ale nie do pobierania ani przeglądania komunikatów, a także do uzyskiwania informacji lub ustawiania atrybutów. Więcej informacji na ten temat zawiera uwaga o składni 8.

Kolejka, która ma QSGDISP (GROUP) , jest specjalnym typem definicji kolejki, którego nie można używać z wywołaniami MQOPEN lub MQPUT1 .

- Lista nazw, w celu:
 - Sprawdź nazwy kolejek na liście (za pomocą wywołania MQINQ).
- Definicja procesu, w celu:
 - Zapytaj o atrybuty procesu (za pomocą wywołania MQINQ).
- Menedżer kolejek, w celu:
 - Sprawdź atrybuty lokalnego menedżera kolejek (za pomocą wywołania MQINQ).

2. Jest on poprawny dla aplikacji, aby otworzyć ten sam obiekt więcej niż jeden raz. Dla każdego otwartego uchwytu zwracany jest inny uchwyt obiektu. Każdy zwracany uchwyt może być użyty dla funkcji, dla których wykonano odpowiednie otwarcie.

3. Jeśli otwierany obiekt jest kolejką, ale nie jest kolejką klastra, wszystkie rozstrzygnięcie nazw w lokalnym menedżerze kolejek odbywa się w czasie wywołania MQOPEN. Może to być jeden lub kilka z następujących elementów dla konkretnego wywołania MQOPEN:

- Rozstrzygnięcie aliasu do nazwy kolejki podstawowej
- Rozstrzygnięcie nazwy lokalnej definicji kolejki zdalnej na nazwę menedżera kolejek zdalnych oraz nazwę, pod którą ta kolejka jest znana w zdalnym menedżerze kolejek
- Rozstrzygnięcie nazwy zdalnego menedżera kolejek na nazwę lokalnej kolejki transmisji

Należy jednak pamiętać, że kolejne wywołania MQINQ lub MQSET dla uchwytu odnoszą się wyłącznie do nazwy, która została otwarta, a nie do obiektu, który ma miejsce po rozstrzygnięcie nazwy.

Na przykład, jeśli otwarto obiekt jest aliasem, atrybuty zwracane przez wywołanie MQINQ są atrybutami aliasu, a nie atrybutami kolejki podstawowej, do której alias jest tłumaczący. Sprawdzanie tłumaczenia nazw nadal jest wykonywane niezależnie od tego, co zostało określone dla parametru *OPTS* w odpowiadaniu na odpowiednie komendy MQOPEN.

Jeśli otwierany obiekt jest kolejką klastra, rozstrzygnięcie nazwy może nastąpić w momencie wywołania MQOPEN lub zostać odroczone do czasu późniejszego. Punkt, w którym występuje rozstrzygnięcie, jest kontrolowany przez opcje OOBND* określone w wywołaniu MQOPEN:

- OOBND0
- OOBNDN
- OOBNDQ

Więcej informacji na temat rozstrzygania nazw kolejek klastra zawiera sekcja [Rozdzielczość nazw](#).

4. Atrybuty obiektu mogą ulec zmianie, gdy aplikacja ma otwarty obiekt. W wielu przypadkach aplikacja tego nie zauważa, ale dla niektórych atrybutów menedżer kolejek oznacza uchwyt, który nie jest już poprawny. Są to:

- Dowolny atrybut, który ma wpływ na rozstrzygnięcie nazwy obiektu. Dotyczy to bez względu na używane opcje otwierania i obejmuje następujące elementy:
 - Zmiana atrybutu *BaseQName* kolejki aliasowej, która jest otwarta.
 - Zmiana atrybutów kolejki produktu *RemoteQName* lub *RemoteQMGrName* dla dowolnego uchwytu, który jest otwarty dla tej kolejki lub dla kolejki, która jest tłumaczona przez tę definicję jako alias menedżera kolejek.
 - Każda zmiana, która powoduje, że aktualnie otwarty uchwyt dla kolejki zdalnej zostanie rozstrzygany w innej kolejce *transmisji* lub w ogóle nie zostanie rozstrzygany. Na przykład może to być:
 - Zmiana atrybutu *XmitQName* w lokalnej definicji kolejki zdalnej, bez względu na to, czy definicja jest używana dla kolejki, czy dla aliasu menedżera kolejek.

Istnieje jeden wyjątek od tego, a mianowicie utworzenie nowej kolejki transmisji. Uchwyt, który mógł zostać rozwiązany do tej kolejki, był obecny podczas otwierania uchwytu, ale został rozstrzygnięty do domyślnej kolejki transmisji, nie jest on niepoprawny.

- Zmiana atrybutu menedżera kolejek produktu *DefXmitQName*. W tym przypadku wszystkie otwarte uchwyt, które zostały rozstrzygnięte do poprzednio nazwanej kolejki (która została rozstrzygnięta tylko dlatego, że była to domyślna kolejka transmisji) są oznaczone jako niepoprawne. Nie ma to wpływu na uchwyt, które zostały przetłumaczone na tę kolejkę z innych przyczyn.
- Atrybut kolejki *Shareability*, jeśli istnieją dwa lub więcej uchwytów, które obecnie zapewniają dostęp OOINPS dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę. Jeśli tak, *wszystkie* uchwyt, które są otwarte dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, są oznaczone jako niepoprawne, niezależnie od otwartych opcji.
- Atrybut kolejki *Usage* dla wszystkich uchwytów otwartych dla tej kolejki lub dla kolejki, która jest tłumaczona do tej kolejki, niezależnie od otwartych opcji.

Jeśli uchwyt jest oznaczony jako niepoprawny, wszystkie kolejne wywołania (inne niż MQCLOSE) korzystające z tego uchwytu nie powiodą się z kodem przyczyny RC2041; aplikacja powinna wywołać wywołanie MQCLOSE (przy użyciu oryginalnego uchwytu), a następnie ponownie otworzyć kolejkę. Wszystkie niezatwierdzone aktualizacje starego uchwytu z poprzednich pomyślnych wywołań nadal mogą być zatwierdzane lub wycofane, zgodnie z wymaganiami logiki aplikacji.

Jeśli zmiana atrybutu spowoduje to, że ma to miejsce, należy użyć specjalnej wersji komendy "force".

5. Menedżer kolejek wykonuje sprawdzenia zabezpieczeń po wywołaniu wywołania MQOPEN w celu sprawdzenia, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma odpowiedni poziom uprawnień, zanim dostęp jest dozwolony. Sprawdzanie uprawnień jest wykonywane na podstawie nazwy otwieranego obiektu, a nie nazwy lub nazw, co spowodowało, że nazwa została rozwiązana.

Jeśli otwierany obiekt jest kolejką modelową, menedżer kolejek wykonuje pełne sprawdzenie zabezpieczeń zarówno dla nazwy kolejki modelowej, jak i nazwy kolejki dynamicznej, która jest tworzona. Jeśli wynikowa kolejka dynamiczna jest otwierana jawnie, dla nazwy kolejki dynamicznej wykonywane jest dalsze sprawdzanie zabezpieczeń zasobów.

6. Kolejka zdalna może być określona na jeden z dwóch sposobów w parametrze *OBJDSC* tego wywołania (patrz pola *ODON* i *ODMN* opisane w sekcji [“MQOD-deskryptor obiektu” na stronie 1176](#)):

- Określając parametr *ODON*, nazwę lokalnej definicji kolejki zdalnej. W takim przypadku produkt *ODMN* odwołuje się do lokalnego menedżera kolejek i może zostać określony jako odstępy.

Sprawdzenie poprawności zabezpieczeń wykonywane przez lokalny menedżer kolejek sprawdza, czy użytkownik jest uprawniony do otwarcia lokalnej definicji kolejki zdalnej.

- Określając parametr *ODON*, nazwę kolejki zdalnej, która jest znana menedżerowi kolejek zdalnych. W tym przypadku *ODMN* jest nazwą zdalnego menedżera kolejek.

Sprawdzenie poprawności zabezpieczeń wykonywane przez lokalny menedżer kolejek sprawdza, czy użytkownik jest uprawniony do wysyłania komunikatów do kolejki transmisji, wynikających z procesu rozstrzygania nazw.

W obu przypadkach:

- Lokalny menedżer kolejek nie wysyła żadnych komunikatów do zdalnego menedżera kolejek w celu sprawdzenia, czy użytkownik jest uprawniony do umieszczania komunikatów w kolejce.
- Gdy komunikat dociera do menedżera kolejek zdalnych, zdalny menedżer kolejek może go odrzucić, ponieważ użytkownik pochodzący z tego komunikatu nie jest autoryzowany.

7. Wywołanie MQOPEN z opcją OOBW ustawia kursor przeglądania, który jest używany z wywołaniami MQGET, które określają uchwyt obiektu i jedną z opcji przeglądania. Pozwala to na skanowanie kolejki bez zmiany jej zawartości. Komunikat, który został znaleziony podczas przeglądania, może zostać później usunięty z kolejki przy użyciu opcji GMMUC.

Wiele kursorów przeglądania może być aktywnych dla pojedynczej aplikacji, wydając kilka żądań MQOPEN dla tej samej kolejki.

8. Do korzystania z list dystrybucyjnych stosuje się następujące uwagi.

- Pola w strukturze MQOD muszą być ustawione w następujący sposób podczas otwierania listy dystrybucyjnej:

- *ODVER* musi mieć wartość *ODVER2* lub większą.
- *ODOT* musi być OTQ.
- Wartość *ODON* musi być pusta lub zawierać łańcuch pusty.
- Wartość *ODMN* musi być pusta lub zawierać łańcuch pusty.
- Wartość *ODREC* musi być większa od zera.
- Jeden z elementów *ODORO* i *ODORP* musi być zerem, a drugi niezerem.
- Nie więcej niż jeden z serwerów *ODRRO* i *ODRRP* może być niezerowy.

- Muszą istnieć rekordy obiektów *ODREC* , które są adresowane zarówno przez produkt *ODORO* , jak i *ODORP* . Rekordy obiektów muszą być ustawione na nazwy kolejek docelowych, które mają zostać otwarte.
- Jeśli jeden z elementów *ODRRO* i *ODRRP* jest niezerowy, muszą istnieć rekordy odpowiedzi *ODREC* . Są one ustawiane przez menedżer kolejek, jeśli wywołanie zostało zakończone z kodem przyczyny RC2136.

Do otwarcia pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej, można użyć komendy version-2 MQOD, upewniając się, że parametr *ODREC* ma wartość zero.

- W parametrze *OPTS* poprawne są tylko następujące opcje otwierania:
 - OOOOUT
 - OOPAS*
 - OOSSET*
 - OOALTU
 - OOFIQ
 - Kolejkami docelowymi na liście dystrybucyjnej mogą być kolejki lokalne, aliasy lub kolejki zdalne, ale nie mogą być kolejkami modelowymi. Jeśli określona jest kolejka modelowa, kolejka ta nie zostanie otwarta, a kod przyczyny RC2057. Nie powoduje to jednak, że inne kolejki na liście zostaną otwarte pomyślnie.
 - Kod zakończenia i parametry kodu przyczyny są ustawione w następujący sposób:
 - Jeśli operacje otwarcia dla kolejek na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, parametry kodu zakończenia i kodu przyczyny zostaną ustawione tak, aby opisywać wspólny wynik. W tym przypadku nie są ustawione rekordy odpowiedzi MQRR (jeśli aplikacja jest udostępniana przez aplikację).

Na przykład, jeśli każde otwarcie powiedzie się, kod zakończenia zostanie ustawiony na wartość CCOK, a kod przyczyny to RCNONE; jeśli każde otwarcie zakończy się niepowodzeniem, ponieważ żadna z kolejek nie istnieje, parametry są ustawiane na CCFAIL i RC2085.

 - Jeśli otwarte operacje dla kolejek na liście dystrybucyjnej nie wszystkie powiodą się lub nie powiodą się w ten sam sposób:
 - Parametr kodu zakończenia jest ustawiony na CCWARN, jeśli co najmniej jedno otwarte powiodło się, oraz do CCFAIL, jeśli wszystkie nie powiodły się.
 - Parametr kodu przyczyny jest ustawiony na wartość RC2136.
 - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.
 - Po pomyślnym otwarciu listy dystrybucyjnej uchwyt *HOBJ* zwracany przez wywołanie może być użyty w kolejnych wywołaniach MQPUT w celu umieszczenia komunikatów w kolejkach na liście dystrybucyjnej, a w wywołaniu MQCLOSE w celu uzyskania dostępu do listy dystrybucyjnej. Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest CONONE.
- Wywołanie MQPUT1 może również zostać użyte do umieszczenia komunikatu na liście dystrybucyjnej; struktura MQOD definiująca kolejki na liście jest określona jako parametr w wywołaniu.
- Każde pomyślne otwarcie miejsca docelowego na liście dystrybucyjnej jest liczone jako *oddzielny* uchwyt podczas sprawdzania, czy aplikacja przekroczyła dozwoloną maksymalną liczbę uchwytów (patrz atrybut menedżera kolejek produktu *MaxHandles*). Jest to prawdą nawet wtedy, gdy co najmniej dwa miejsca docelowe na liście dystrybucyjnej są rzeczywiście rozstrzygane do tej samej kolejki fizycznej. Jeśli wywołanie MQOPEN lub MQPUT1 dla listy dystrybucyjnej spowodowałoby, że liczba uchwytów używanych przez aplikację przekroczy *MaxHandles*, wywołanie nie powiedzie się i zostanie zakodowany kod przyczyny RC2017.
 - Każdy otwarty cel, który został pomyślnie otwarty, ma wartość atrybutu *OpenOutputCount* zwiększoną o jeden. Jeśli co najmniej dwa miejsca docelowe na liście dystrybucyjnej są w rzeczywistości rozstrzygane do tej samej kolejki fizycznej, atrybut *OpenOutputCount* jest

zwiększany o liczbę miejsc docelowych znajdujących się na liście dystrybucyjnej, które są rozstrzygane do tej kolejki.

- Każda zmiana w definicjach kolejek, które spowodowałyby, że uchwyt stał się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana ścieżki rozdzielczej), nie powoduje, że uchwyt listy dystrybucyjnej staje się niepoprawny. Jednak powoduje to niepowodzenie tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnych wywoławczych wywołania MQPUT.
- Jest ona poprawna dla listy dystrybucyjnej, która może zawierać tylko jedno miejsce docelowe.

9. Poniższe uwagi dotyczą korzystania z kolejek klastra.

- Gdy kolejka klastra jest otwierana po raz pierwszy, a lokalny menedżer kolejek nie jest pełnym menedżerem kolejek repozytorium, lokalny menedżer kolejek uzyskuje informacje na temat kolejki klastra z pełnego menedżera kolejek repozytorium. Gdy sieć jest zajęta, lokalny menedżer kolejek może zająć kilka sekund, aby otrzymać potrzebne informacje z menedżera kolejek repozytorium. W wyniku tego aplikacja wywołująca wywołanie MQOPEN może mieć do 10 sekund oczekiwania przed zwróceniem sterowania z wywołania MQOPEN. Jeśli lokalny menedżer kolejek nie otrzyma w tym czasie potrzebnych informacji na temat kolejki klastra, wywołanie nie powiedzie się i zostanie wyświetlony kod przyczyny RC2189.

- Gdy kolejka klastra jest otwierana i istnieje wiele instancji kolejki w klastrze, instancja ta jest rzeczywiście otwarta, zależy od opcji określonych w wywołaniu MQOPEN:

– Jeśli podane opcje zawierają dowolną z następujących opcji:

- OOBWR
- POINPQ
- OOINPX
- OOINPS
- OOSSET

Instancja kolejki klastra otwarta jest wymagana jako instancja lokalna. Jeśli nie istnieje lokalna instancja kolejki, wywołanie MQOPEN nie powiedzie się.

– Jeśli podane opcje nie zawierają żadnego z powyższych, ale należy uwzględnić jedno lub oba z następujących elementów:

- OOINQ
- OOOOUT

Instancja otwarta jest instancją lokalną, jeśli istnieje jedna instancja, a instancja zdalna w przeciwnym razie. Instancja wybrana przez menedżer kolejek może jednak zostać zmieniona przez wyjście obciążenia klastra (jeśli istnieje).

Więcej informacji na temat kolejek klastra zawiera sekcja [Kolejki klastrów](#).

10. Aplikacje uruchomione przez monitor wyzwalacza są przekazywane do nazwy kolejki powiązanej z aplikacją, gdy aplikacja jest uruchomiona. Tę nazwę kolejki można określić w parametrze *OBJDSC*, aby otworzyć kolejkę. Szczegółowe informacje można znaleźć w opisie struktury MQTMC.
11. W systemie IBM aplikacje działające w trybie zgodności są automatycznie połączone z menedżerem kolejek przy użyciu pierwszego wywołania MQOPEN wydanego przez aplikację (jeśli aplikacja nie nawiązała jeszcze połączenia z menedżerem kolejek przy użyciu wywołania MQCONN).

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilewanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

Aplikacje, które nie działają w trybie zgodności, muszą wywołać wywołanie MQCONN lub MQCONNX w celu jawnego nawiązania połączenia z menedżerem kolejek przed użyciem wywołania MQOPEN w celu otwarcia obiektu.

12. W przypadku korzystania z opcji OORLOQ kolejka lokalna jest już zwracana, gdy otwarta jest kolejka lokalna, alias lub kolejka modelowa, ale nie jest to w przypadku, gdy na przykład otwierana jest kolejka zdalna lub kolejka nielokalnego klastra; nazwy ResolvedQName i ResolvedQMgrsą wprowadzane razem z nazwą RemoteQName i RemoteQMgrznalezionym w definicji kolejki zdalnej lub podobnie z wybraną kolejką klastra zdalnego. Jeśli wartość OORLOQ jest określona podczas otwierania, na przykład w kolejce zdalnej, ResolvedQName będzie teraz kolejką transmisji, do której będą umieszczane komunikaty. Nazwa ResolvedQMgrzostanie wprowadzona wraz z nazwą lokalnego menedżera kolejek udostępniającego kolejkę transmisji. Jeśli użytkownik jest uprawniony do przeglądania, wprowadzania danych wejściowych lub wyjściowych w kolejce, mają one uprawnienia wymagane do określenia tej flagi w wywołaniu MQOPEN. Nie jest wymagane żadne uprawnienie specjalne.

Parametry

Wywołanie MQOPEN ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwył połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

OBJDSC (MQOD)-wejście/wyjście

Deskryptor obiektu.

Jest to struktura identyfikująca obiekt, który ma zostać otwarty. Szczegółowe informacje znajdują się w sekcji [“MQOD-deskryptor obiektu”](#) na stronie 1176 .

Jeśli pole *ODON* w parametrze *OBJDSC* jest nazwą kolejki modelowej, dynamiczna kolejka lokalna jest tworzony z atrybutami kolejki modelowej; dzieje się tak niezależnie od otwartych opcji określonych przez parametr *OPTS* . Kolejne operacje przy użyciu *HOBJ* zwróconego przez wywołanie MQOPEN są wykonywane w nowej kolejce dynamicznej, a nie w kolejce modelowej. Jest to prawda nawet w przypadku wywołań MQINQ i MQSET. Nazwa kolejki modelowej w parametrze *OBJDSC* jest zastępowana nazwą utworzonej kolejki dynamicznej. Typ kolejki dynamicznej jest określany na podstawie wartości atrybutu *DefinitionType* kolejki modelowej (patrz [“Atrybuty dla kolejek”](#) na stronie 1395). Informacje na temat opcji zamykania, które mają zastosowanie do kolejek dynamicznych, zawiera opis wywołania MQCLOSE.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje sterujące działaniem komendy MQOPEN.

Należy podać co najmniej jedną z następujących opcji:

- OOBWR
- OOINP* (tylko jedno z nich)
- OOINQ
- OOOOT
- OOSET
- OORLQ

Inne opcje mogą być określone zgodnie z wymaganiami. Jeśli wymagana jest więcej niż jedna opcja, można dodać wartości (nie należy dodawać tej samej stałej więcej niż raz). Podane kombinacje nie są poprawne; wszystkie pozostałe kombinacje są poprawne. Dozwolone są tylko opcje, które mają zastosowanie do typu obiektu określonego przez program *OBJDSC* (patrz [Poprawne opcje MQOPEN dla każdego typu kolejki](#)).

Opcje dostępu: Następujące opcje sterują typem operacji, które mogą być wykonywane na obiekcie:

POINPQ

Otwieranie kolejki w celu pobierania komunikatów za pomocą wartości domyślnej zdefiniowanej przez kolejkę.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Typ dostępu jest współużytkowany lub na wyłączność, w zależności od wartości atrybutu kolejki produktu *DefInputOpenOption*. Szczegółowe informacje zawiera sekcja [“Atrybuty dla kolejek” na stronie 1395](#).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

OOINPS

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest aktualnie otwarta przez tę lub inną aplikację z OOINPS, ale kończy się niepowodzeniem z kodem przyczyny RC2042, jeśli kolejka jest obecnie otwarta z OOINPX.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

OOINPX

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie nie powiodło się z kodem przyczyny RC2042, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla danych wejściowych dowolnego typu (OOINPS lub OOINPX).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

Do tych opcji mają zastosowanie następujące uwagi:

- Można określić tylko jedną z tych opcji.
- Wywołanie MQOPEN z jedną z tych opcji może się powieść nawet wtedy, gdy atrybut kolejki *InhibitGet* jest ustawiony na wartość QAGETI (choćby kolejne wywołania MQGET nie powiodą się, gdy atrybut zostanie ustawiony na tę wartość).
- Jeśli kolejka jest zdefiniowana jako niewspółużytkowalna (czyli atrybut kolejki *Shareability* ma wartość QANSHR), próby otwarcia kolejki na potrzeby współużytkowanego dostępu są traktowane jako próby otwarcia kolejki z wyłącznym dostępem.
- Jeśli kolejka aliasowa jest otwierana przy użyciu jednej z tych opcji, test wyłącznego użycia (lub dla tego, czy inna aplikacja ma wyłączne użycie) jest dla kolejki podstawowej, do której alias jest tłumaczący.
- Te opcje nie są poprawne, jeśli *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej przez aliasing menedżera kolejek jest nazwą lokalnego menedżera kolejek.

OBRW

Otwórz kolejkę, aby przeglądać komunikaty.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET z jedną z następujących opcji:

- GMBRWF
- GMBRWN

- GMBRWC

Jest to dozwolone nawet wtedy, gdy kolejka jest obecnie otwarta dla OOINPX. Wywołanie MQOPEN z opcją OOBRW tworzy kursor przeglądania i umieszcza je logicznie przed pierwszym komunikatem w kolejce. W celu uzyskania dalszych informacji należy zapoznać się z polem *GMOPT* opisanym w sekcji “MQGMO-Opcje Get-message” na stronie 1089.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Nie jest również poprawna, jeśli *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej przez aliasing menedżera kolejek jest nazwą lokalnego menedżera kolejek.

OOOUT

Otwieranie kolejki w celu umieszczania komunikatów lub łańcucha tematu lub tematu w celu publikowania komunikatów.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQPUT.

Wywołanie MQOPEN z tą opcją może się powieść nawet wtedy, gdy atrybut kolejki *InhibitPut* jest ustawiony na wartość QAPUTI (choćby kolejne wywołania MQPUT nie powiodą się, gdy atrybut zostanie ustawiony na tę wartość).

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych i tematów.

OOINQ

Otwórz obiekt, aby uzyskać dostęp do atrybutów.

Kolejka, lista nazw, definicja procesu lub menedżer kolejek są otwierane w celu użycia z kolejnymi wywołaniami MQINQ.

Ta opcja jest poprawna dla wszystkich typów obiektów innych niż listy dystrybucyjne. Wartość ta nie jest poprawna, jeśli *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej przez aliasing menedżera kolejek jest nazwą lokalnego menedżera kolejek.

OOSSET

Otwieranie kolejki w celu ustawienia atrybutów.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQSET.

Ta opcja jest poprawna dla wszystkich typów kolejek innych niż listy dystrybucyjne. Wartość ta nie jest poprawna, jeśli *ODMN* jest nazwą lokalnej definicji kolejki zdalnej. Jest to prawda, nawet jeśli wartość atrybutu *RemoteQMGrName* w definicji lokalnej kolejki zdalnej używanej przez aliasing menedżera kolejek jest nazwą lokalnego menedżera kolejek.

Opcje powiązania: Następujące opcje mają zastosowanie, gdy otwierany obiekt jest kolejką klastra; te opcje sterują powiązaniem uchwytu kolejki z instancją kolejki klastra:

OOBND

Powiązanie uchwyt z miejscem docelowym, gdy kolejka jest otwierana.

Spowoduje to, że lokalny menedżer kolejek powiąże uchwyt kolejki z instancją kolejki docelowej, gdy kolejka jest otwarta. W wyniku tego wszystkie komunikaty umieszczone przy użyciu tego uchwytu są wysyłane do tej samej instancji kolejki docelowej, a także do tej samej trasy.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

OOBNDN

Nie należy wiązać się z konkretnym miejscem docelowym.

Spowoduje to zatrzymanie menedżera kolejek lokalnych, który powiąże uchwyt kolejki z instancją kolejki docelowej. W rezultacie kolejne wywołania MQPUT korzystające z tego uchwytu mogą spowodować, że komunikaty będą wysyłane do *różnych* instancji kolejki docelowej lub są wysyłane do tej samej instancji, ale na różne trasy. Umożliwia również zmianę instancji

wybranej później przez lokalny menedżer kolejek, menedżer kolejek zdalnych lub agent kanału komunikatów (MCA), zgodnie z warunkami sieciowymi.

Uwaga: Aplikacje klienckie i serwerowe, które muszą wymieniać *serię* komunikatów w celu zakończenia transakcji, nie powinny używać OOBNDN (lub OOBNDQ, gdy *DefBind* ma wartość BNDNOT), ponieważ kolejne komunikaty z serii mogą być wysyłane do różnych instancji aplikacji serwera.

Jeśli dla kolejki klastra określono opcję OOBRW lub jedną z opcji OOINP*, menedżer kolejek jest zmuszony do wybrania lokalnej instancji kolejki klastra. Oznacza to, że powiązanie uchwytu kolejki jest stałe, nawet jeśli określono wartość OOBNDN.

Jeśli wartość OOINQ jest określona z OOBNDN, kolejne wywołania MQINQ korzystające z tego uchwytu mogą zapytać o różne instancje kolejki klastra, chociaż zwykle wszystkie instancje mają takie same wartości atrybutów.

Wartość OOBNDN jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja jest określona dla kolejki, która nie jest kolejką klastra, opcja jest ignorowana.

OOBNDQ

Użyj domyślnego powiązania dla kolejki.

Powoduje to, że lokalny menedżer kolejek powiąże uchwyt kolejki w sposób zdefiniowany przez atrybut kolejki *DefBind*. Wartością tego atrybutu jest BNDOPN lub BNDNOT.

Wartość OOBNDQ jest wartością domyślną, jeśli nie określono OOBNDQ i OOBNDN.

OOBNDQ jest zdefiniowane w dokumentacji programu pomocy. Opcja ta nie jest przeznaczona dla żadnej z dwóch pozostałych opcji wiązania, ale ponieważ jej wartość jest równa zero, nie można jej wykryć.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

OOSAVA

Zapisz kontekst podczas pobierania komunikatu.

Informacje o kontekście są powiązane z tym uchwytym kolejki. Te informacje są ustawiane na podstawie kontekstu dowolnego komunikatu pobranego przy użyciu tego uchwytu. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Te informacje o kontekście mogą być przekazywane do komunikatu, który jest później umieszczany w kolejce przy użyciu wywołań MQPUT lub MQPUT1. Zapoznaj się z opcjami PMPASI i PMPASA opisanymi w sekcji [“MQPMO-Put-message, opcje”](#) na stronie 1191.

Dopóki komunikat nie zostanie pomyślnie pobrany, nie można przekazać kontekstu do komunikatu umieszczanego w kolejce.

Komunikat pobrany przy użyciu jednej z opcji przeglądania GMBRW* nie ma zapisanych informacji o kontekście (choć pola kontekstu w parametrze MSGDSC są ustawione po przeglądaniu).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest ona poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Należy określić jedną z opcji OOINP*.

OOPASI

Zezwalaj na przekazanie kontekstu tożsamości.

Umożliwia to określenie opcji PMPASI w parametrze PMO, gdy komunikat jest umieszczany w kolejce. Pozwala to na przesłanie informacji o kontekście tożsamości z kolejki wejściowej, która została otwarta z opcją OOSAVA. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji [Kontekst komunikatu](#) i [Informacje o sterowaniu kontekstem](#).

Należy określić opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

OOPASA

Zezwól na przekazanie całego kontekstu.

Pozwala to na określenie opcji PMPASA w parametrze *PMO* , gdy komunikat jest umieszczany w kolejce. To daje komunikat o tożsamości i kontekście pochodzenia z kolejki wejściowej, która została otwarta z opcją OOSAVA. Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Opcja ta oznacza OOPASI, które nie są w związku z tym określone. Należy określić opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

OOSSETI

Zezwalaj na ustawienie kontekstu tożsamości.

Umożliwia to określenie opcji PMSETI w parametrze *PMO* , gdy komunikat jest umieszczany w kolejce. To daje komunikat do informacji o kontekście tożsamości zawartych w parametrze *MSGDSC* określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Opcja ta oznacza OOPASI, które nie są w związku z tym określone. Należy określić opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

OOSSETA

Zezwól na ustawienie całego kontekstu.

Umożliwia to określenie opcji PMSETA w parametrze *PMO* , gdy komunikat jest umieszczany w kolejce. To daje komunikat do informacji o kontekście tożsamości i pochodzenia zawartych w parametrze *MSGDSC* określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu można znaleźć w sekcji Kontekst komunikatu i Informacje o sterowaniu kontekstem.

Ta opcja oznacza następujące opcje, które nie muszą być określone:

- OOPASI
- OOPASA
- OOSSETI

Należy określić opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych.

Inne opcje: Następujące opcje kontrolują sprawdzanie autoryzacji i to, co się dzieje, gdy menedżer kolejek jest wygaszany:

OOALTU

Sprawdź poprawność z określonym identyfikatorem użytkownika.

Oznacza to, że pole *ODAU* w parametrze *OBJDSC* zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności wywołania MQOPEN. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy *ODAU* jest autoryzowany do otwarcia obiektu przy użyciu określonych opcji dostępu, niezależnie od tego, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma do tego uprawnienia. Nie dotyczy to jednak żadnych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym aplikacja jest uruchomiona.

Ta opcja jest poprawna dla wszystkich typów obiektów.

OOFIQ

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

Ta opcja wymusza, że wywołanie MQOPEN nie powiedzie się, jeśli menedżer kolejek jest w stanie wygaszania.

Ta opcja jest poprawna dla wszystkich typów obiektów.

OORLQ

Wprowadź nazwę kolejki lokalnej, która została otwarta.

Ta opcja określa, że wartość ResolvedQName w strukturze MQOD (jeśli jest dostępna) powinna zostać wprowadzona wraz z nazwą kolejki lokalnej, która została otwarta. Podobnie nazwa ResolvedQMgr zostanie wprowadzona z nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną.

Poprawne opcje MQOPEN dla każdego typu kolejki

Opcja	Alias ("1" na stronie 1357)	Lokalne i modelowe	Zdalny	Klaster inny niż lokalny	Lista dystrybucyjna	Temat
POINPQ	✓	✓	-	-	-	-
OOINPS	✓	✓	-	-	-	-
OOINPX	✓	✓	-	-	-	-
OOBRW	✓	✓	-	-	-	-
OOOUT	✓	✓	✓	✓	✓	✓
OOINQ	✓	✓	"2" na stronie 1357	✓	-	-
OOSET	✓	✓	"2" na stronie 1357	-	-	-
OOBNDQ ("3" na stronie 1357)	✓	✓	✓	✓	✓	-
OOBNDN ("3" na stronie 1357)	✓	✓	✓	✓	✓	-
OOBNDQ ("3" na stronie 1357)	✓	✓	✓	✓	✓	-
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	"5" na stronie 1357
OOPASA	✓	✓	✓	✓	✓	"5" na stronie 1357
OOSETI	✓	✓	✓	✓	✓	"5" na stronie 1357
OOSETA	✓	✓	✓	✓	✓	"5" na stronie 1357
OOALTU	✓	✓	✓	✓	✓	✓
OOFIQ	✓	✓	✓	✓	✓	✓

Opcja	Alias ("1" na stronie 1357)	Lokalne i modelowe	Zdalny	Klaster inny niż lokalny	Lista dystrybucyjna	Temat
OORLQ	✓	✓	✓	✓	-	-

Uwagi:

1. Ważność opcji dla aliasów zależy od poprawności opcji kolejki, do której jest rozstrzygany alias.
2. Ta opcja jest poprawna tylko w przypadku lokalnej definicji kolejki zdalnej.
3. Ta opcja może być określona dla dowolnego typu kolejki, ale jest ignorowana, jeśli kolejka nie jest kolejką klastra.
4. Ten atrybut jest ignorowany dla tematu.
5. Atrybuty te mogą być używane z tematem, ale mają wpływ tylko na kontekst ustawiony dla zachowanego komunikatu, a nie na pola kontekstu wysyłane do żadnego subskrybenta.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt obiektu.

Ten uchwyt reprezentuje dostęp, który został utworzony dla obiektu. Musi być ona określona w kolejnych wywołaniach kolejkowania komunikatów, które działają na obiekcie. Traci ona ważność po wywołaniu wywołania MQCLOSE lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, kończy działanie.

Zakres uchwytu jest ograniczony do najmniejszej jednostki równoległe przetwarzanie obsługiwane przez platformę, na której działa aplikacja; uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której została wywołana wywołanie MQOPEN:

- W systemie IBM izakres uchwytu jest zadaniem wydającym wywołanie.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

Deklaracja RPG

```

C*.1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQOPEN(HCONN : OBJDSC : OPTS :
C                                HOBJ : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```

D*.1.....2.....3.....4.....5.....6.....7..
DMQOPEN      PR          EXTPROC('MQOPEN')
D* Connection handle
D HCONN              10I 0 VALUE
D* Object descriptor
D OBJDSC              468A
D* Options that control the action of MQOPEN
D OPTS              10I 0 VALUE
D* Object handle
D HOBJ              10I 0
D* Completion code
```

D CMPCOD	10I 0
D* Reason code qualifying CMPCOD	
D REASON	10I 0

MQPUT-umieszczanie komunikatu

Wywołanie MQPUT umieszcza komunikat w kolejce, liście dystrybucyjnej lub w temacie. Kolejka, lista dystrybucyjna lub temat muszą być już otwarte.

- [“Składnia” na stronie 1358](#)
- [“Użycie notatek” na stronie 1358](#)
 - [“Tematy” na stronie 1358](#)
 - [“MQPUT i MQPUT1” na stronie 1359](#)
 - [“Kolejki docelowe” na stronie 1359](#)
 - [“Lista dystrybucyjna” na stronie 1360](#)
 - [“Nagłówki” na stronie 1361](#)
 - [“Buforuj” na stronie 1362](#)
- [“Parametry” na stronie 1362](#)
- [“Deklaracja RPG” na stronie 1367](#)

Składnia

MQPUT (*HCONN, HOBJ, MSGDSC, PMO, BUFLN, BUFFER, CMPCOD, REASON*)

Użycie notatek

Tematy

Następujące uwagi mają zastosowanie w przypadku korzystania z tematów:

1. W przypadku użycia komendy MQPUT do publikowania komunikatów w temacie, w którym co najmniej jeden subskrybent tego tematu nie może zostać nadany publikacji z powodu problemu z kolejką subskrybentów (na przykład jest pełna), kod przyczyny zwrócony do wywołania MQPUT i zachowanie dostarczania zależy od ustawienia atrybutów PMSGDLV lub NPMSGDLV w temacie TOPIC. Należy zwrócić uwagę, że dostarczenie publikacji do kolejki niedostarczanej poczty w przypadku określenia parametru RODLQ lub odrzucenie komunikatu w przypadku określenia parametru RODISC, jest uważane za pomyślne dostarczenie komunikatu. Jeśli żadna z tych publikacji nie zostanie dostarczona, komenda MQPUT zwróci wartość RC2502. Może się to zdarzyć w następujących przypadkach:
 - Komunikat jest publikowany w TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALL, a każda subskrypcja (trwała lub nie) ma kolejkę, która nie może odbierać publikacji.
 - Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLDUR, a subskrypcja trwała ma kolejkę, która nie może odbierać publikacji.

Komenda MQPUT może zwracać wartość RCNONE, nawet jeśli publikacje nie mogą być dostarczane do niektórych subskrybentów w następujących przypadkach:

- Komunikat jest publikowany w TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLAVAIL, a każda subskrypcja, trwała lub nie, ma kolejkę, która nie może odbierać publikacji.
- Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALLDUR, a subskrypcja nietrwała ma kolejkę, która nie może odbierać publikacji.

2. Jeśli nie ma subskrybentów w używanym temacie, opublikowany komunikat nie jest wysyłany do żadnej kolejki i jest usuwany. Nie ma znaczenia, czy ten komunikat jest trwały, czy nietrwały, czy ma nieograniczony limit czasu utraty ważności, czy niewielki czas utraty ważności. Jest on nadal usuwany, jeśli nie ma subskrybentów. Wyjątkiem jest sytuacja, w której komunikat ma zostać zachowany. W takim przypadku, mimo że nie jest on wysyłany do kolejek subskrybentów, zostanie on zapisany w temacie, który ma zostać dostarczony do nowych subskrypcji lub do wszystkich subskrybentów, którzy poproszli o zachowane publikacje za pomocą komendy MQSUBRQ.

MQPUT i MQPUT1

Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. W zależności od okoliczności używane jest wywołanie do użycia.

- Wywołanie MQPUT powinno być używane, gdy wiele komunikatów ma być umieszczonych w *tej samej* kolejce.

Wywołanie MQOPEN z opcją OOOUT jest wysyłane jako pierwsze, po którym następuje jedna lub większa liczba żądań MQPUT w celu dodania komunikatów do kolejki. W końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1.

- Wywołanie MQPUT1 powinno być używane, gdy tylko *jeden* komunikat ma zostać umieszczony w kolejce.

Wywołanie to enkapsuluje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wysłane.

Kolejki docelowe

Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są następujące warunki. Niektóre warunki mają zastosowanie zarówno do lokalnych, jak i zdalnych kolejek docelowych; inne warunki mają zastosowanie tylko do kolejek zdalnych miejsc docelowych.

Warunki dla lokalnych i zdalnych kolejek docelowych

- Wszystkie wywołania MQPUT znajdują się w tej samej jednostce pracy lub żaden z nich nie znajduje się w obrębie jednostki pracy.

Gdy komunikaty są umieszczane w określonej kolejce w ramach pojedynczej jednostki pracy, komunikaty z innych aplikacji mogą być przeplatane z kolejnością komunikatów w kolejce.

- Wszystkie wywołania MQPUT są wykonywane przy użyciu tego samego uchwytu obiektu *HOBJ*.

W niektórych środowiskach kolejność komunikatów jest również zachowywana, gdy używane są różne uchwyty obiektów, pod warunkiem, że wywołania są wykonywane z tej samej aplikacji. Znaczenie "tej samej aplikacji" jest określone przez środowisko:

– W systemie IBM i aplikacja jest zadaniem.

- Wszystkie komunikaty mają ten sam priorytet.

Dodatkowe warunki dla kolejek zdalnych miejsc docelowych

- Istnieje tylko jedna ścieżka od wysyłającego menedżera kolejek do docelowego menedżera kolejek.

Jeśli istnieje możliwość, że niektóre komunikaty w sekwencji mogą znajdować się w innej ścieżce (na przykład z powodu rekonfiguracji, równoważenia ruchu lub wyboru ścieżki w zależności od wielkości komunikatu), nie można zagwarantować kolejności komunikatów w docelowym menedżerze kolejek.

- Komunikaty nie są tymczasowo umieszczane w kolejkach niedostarczonych komunikatów w menedżerach kolejek nadawczych, pośrednich i docelowych.

Jeśli co najmniej jeden komunikat jest tymczasowo umieszczany w kolejce niedostarczonych komunikatów (na przykład, ponieważ kolejka transmisji lub kolejka docelowa jest tymczasowo pełna), komunikaty mogą być odbierane w kolejce docelowej poza kolejnością.

- Komunikaty są albo wszystkie trwałe, albo wszystkie nietrwałe.

Jeśli kanał na trasie między menedżerami kolejek wysyłających i docelowych ma atrybut *CDNPM* ustawiony na wartość *NPFAST*, nietrwałe komunikaty mogą przeskoczyć przed trwałymi komunikatami, co powoduje, że porządek komunikatów trwałych względem nietrwałych komunikatów nie jest zachowany. Jednak kolejność komunikatów trwałych względem siebie oraz komunikatów nietrwałych, względem siebie wzajemnie, jest zachowywana.

Jeśli te warunki nie są spełnione, grupy komunikatów mogą być używane do zachowania kolejności komunikatów, ale należy pamiętać, że wymaga to zarówno aplikacji wysyłającej, jak i odbierającej w celu korzystania z obsługi grupowania komunikatów. Więcej informacji na temat grup komunikatów zawiera sekcja:

- Pole *MDMFL* w strukturze *MQMD*
- Opcja *PMLOGO* w *MQPMO*
- Opcja *GMLOGO* w *MQGMO*

Lista dystrybucyjna

Do korzystania z list dystrybucyjnych stosuje się następujące uwagi.

1. Komunikaty mogą być umieszczane na liście dystrybucyjnej przy użyciu programu *MQPMO* w wersji *version-1* lub *version-2*. Jeśli używany jest produkt *MQPMO* w wersji *version-1* (lub *version-2* *MQPMO* z wartością *PMREC* równą zero), aplikacja nie może dostarczyć rekordów komunikatów ani rekordów odpowiedzi. Oznacza to, że nie będzie możliwe zidentyfikowanie kolejek, w których występują błędy, jeśli komunikat zostanie pomyślnie wysłany do niektórych kolejek na liście dystrybucyjnej, a nie do innych.

Jeśli aplikacja udostępniła rekordy komunikatów lub rekordy odpowiedzi, pole *PMVER* musi być ustawione na wartość *PMVER2*.

Narzędzie *MQPMO* *version-2* może być również używane do wysyłania komunikatów do jednej kolejki, która nie znajduje się na liście dystrybucyjnej, upewniając się, że parametr *PMREC* ma wartość zero.

2. Kod zakończenia i parametry kodu przyczyny są ustawione w następujący sposób:

- Jeśli wszystkie operacje umieszczania w kolejkach na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, to kod zakończenia i parametry kodu przyczyny zostaną ustawione w taki sposób, aby opisywać wspólny wynik. W tym przypadku nie są ustawione rekordy odpowiedzi *MQRR* (jeśli aplikacja jest udostępniana przez aplikację).

Na przykład, jeśli każde wykonanie powiedzie się, kod zakończenia zostanie ustawiony na wartość *CCOK*, a kod przyczyny to *RCNONE*. Jeśli każde wykonanie nie powiedzie się, ponieważ wszystkie kolejki są zablokowane dla operacji *put*, parametry są ustawiane na *CCFAIL* i *RC2051*.

- Jeśli operacje umieszczania w kolejkach na liście dystrybucyjnej nie wszystkie powiodą się lub nie powiodą się w ten sam sposób:
 - Parametr kodu zakończenia jest ustawiony na *CCWARN*, jeśli co najmniej jeden został pomyślnie zakończony, oraz do *CCFAIL*, jeśli wszystkie nie powiodły się.
 - Parametr kodu przyczyny jest ustawiony na wartość *RC2136*.
 - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.

Jeśli operacja *put* dla miejsca docelowego nie powiedzie się, ponieważ otwarcie dla tego miejsca docelowego nie powiodło się, pola w rekordzie odpowiedzi są ustawione na *CCFAIL* i *RC2137*; , które miejsce docelowe jest dołączone do produktu *PMIDC*.

3. Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę lokalną, komunikat jest umieszczany w tej kolejce w normalnej formie (czyli nie jako komunikat z listą dystrybucyjną). Jeśli więcej niż jedno miejsce docelowe jest tłumaczone na tę samą kolejkę lokalną, w kolejce dla każdego z tych miejsc docelowych umieszczany jest jeden komunikat.

Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę zdalną, komunikat jest umieszczany w odpowiedniej kolejce transmisji. Jeśli kilka miejsc docelowych jest rozstrzyganych

w tej samej kolejce transmisji, w kolejce transmisji może zostać umieszczony pojedynczy komunikat z listą dystrybucyjną, który zawiera te miejsca docelowe, nawet jeśli te miejsca docelowe nie były umieszczone obok listy miejsc docelowych udostępnionych przez aplikację. Można to jednak zrobić tylko wtedy, gdy kolejka transmisji obsługuje komunikaty listy dystrybucyjnej (patrz atrybut kolejki *DistLists* opisany w sekcji „Atrybuty dla kolejek” na stronie 1395).

Jeśli kolejka transmisji nie obsługuje list dystrybucyjnych, jedna kopia komunikatu w zwykłej formie jest umieszczana w kolejce transmisji dla każdego miejsca docelowego, które korzysta z tej kolejki transmisji.

Jeśli lista dystrybucyjna z danymi komunikatu aplikacji jest zbyt duża dla kolejki transmisji, komunikat z listą dystrybucyjną jest dzielony na mniejsze komunikaty listy dystrybucyjnej, z których każda zawiera mniej miejsc docelowych. Jeśli dane komunikatu aplikacji tylko wpisują się do kolejki, komunikaty listy dystrybucyjnej nie mogą być używane w ogóle, a menedżer kolejek generuje jedną kopię komunikatu w postaci normalnej dla każdego miejsca docelowego, które korzysta z tej kolejki transmisji.

Jeśli różne miejsca docelowe mają inny priorytet komunikatu lub trwałość komunikatu (może to wystąpić, gdy aplikacja określa PRQDEF lub PEQDEF), komunikaty nie są przechowywane w tej samej komunikacie listy dystrybucyjnej. Zamiast tego menedżer kolejek generuje tyle komunikatów listy dystrybucyjnej, które są niezbędne do uwzględnienia różnych wartości priorytetu i trwałości.

4. Umieszczenie na liście dystrybucyjnej może spowodować:

- Pojedynczy komunikat z listą dystrybucyjną, lub
- Liczba mniejszych komunikatów listy dystrybucyjnej, lub
- Mieszanie komunikatów listy dystrybucyjnej i zwykłych komunikatów, lub
- Tylko komunikaty normalne.

To, które z poprzednich wystąpień zależy od tego, czy:

- Miejsca docelowe na liście są lokalne, zdalne lub mieszane.
- Miejsca docelowe mają ten sam priorytet komunikatu i trwałość komunikatu.
- Kolejki transmisji mogą zawierać komunikaty listy dystrybucyjnej.
- Maksymalna długość kolejek transmisji jest wystarczająco duża, aby pomieścić komunikat w postaci listy dystrybucyjnej.

Jednak niezależnie od tego, który z powyższych zdarzeń występuje, każdy komunikat *fizyczny* (czyli każdy komunikat normalny lub komunikat listy dystrybucyjnej będący wynikiem umieszczenia) jest wyświetlany jako tylko *jeden* komunikat, gdy:

- Sprawdzanie, czy aplikacja przekroczyła dozwoloną maksymalną liczbę komunikatów w jednostce pracy (patrz atrybut menedżera kolejek produktu *MaxUncommittedMsgs*).
- Sprawdzanie, czy warunki wyzwiania są spełnione.
- Zwiększ głębokość kolejki i sprawdź, czy maksymalna głębokość kolejki została przekroczona.

5. Każda zmiana w definicjach kolejek, które spowodowałyby, że uchwyt stał się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana ścieżki rozdzielczej), nie powoduje, że uchwyt listy dystrybucyjnej staje się niepoprawny. Jednak powoduje to niepowodzenie tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnych wywołaniach MQPUT.

Nagłówki

Jeśli na początku danych komunikatu aplikacji zostanie umieszczony komunikat zawierający jedną lub większą liczbę struktur nagłówka produktu IBM MQ, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka w celu sprawdzenia, czy są one poprawne. Jeśli menedżer kolejek wykryje błąd, wywołanie nie powiedzie się i zostanie zwrócony odpowiedni kod przyczyny. Przeprowadzone kontrole różnią się w zależności od konkretnych struktur, które są obecne. Ponadto kontrole są wykonywane tylko wtedy, gdy w wywołaniu MQPUT lub MQPUT1 jest używany deskryptor MQMD w wersji version-2 lub późniejszej; sprawdzenia nie są wykonywane, jeśli używany jest deskryptor MQMD w wersji version-1, nawet jeśli na początku danych komunikatu aplikacji znajduje się tabela MQMDE.

Następujące struktury nagłówka produktu IBM MQ są całkowicie sprawdzane przez menedżera kolejek: MQDH, MQMDE.

W przypadku innych struktur nagłówka produktu IBM MQ menedżer kolejek wykonuje pewne sprawdzanie poprawności, ale nie sprawdza każdego pola. Struktury, które nie są obsługiwane przez lokalny menedżer kolejek i struktury po pierwszym komunikacie MQDLH w komunikacie, nie są sprawdzane.

Oprócz ogólnych sprawdzeń w polach w strukturach produktu IBM MQ muszą być spełnione następujące warunki:

- Struktura IBM MQ nie może być podzielona na dwa lub więcej segmentów-struktura musi być w całości zawarta w jednym segmencie.
- Suma długości struktur w komunikacie PCF musi być równa długości określonej za pomocą parametru *BUFLen* w wywołaniu MQPUT lub MQPUT1. Komunikat PCF to komunikat, który ma jedną z następujących nazw formatów:
 - FMADMN
 - FMEVNT
 - FMPCF
- Struktury IBM MQ nie mogą być obcinane, z wyjątkiem następujących sytuacji, w których dozwolone są obcinane struktury:
 - Komunikaty, które są komunikatami raportu.
 - Komunikaty PCF.
 - Komunikaty zawierające strukturę MQDLH. (Struktury *następujące* pierwsze wywołanie MQDLH może zostać obcięte; struktury poprzedzające obiekt MQDLH nie mogą być obcinane).

Buforuj

Parametr *BUFFER* przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch; ograniczenie to ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr powinien zostać zadeklarowany jako struktura, albo jako pole w zbiorze fizycznym. Spowoduje to zwiększenie maksymalnej długości do około 32 kB.

Parametry

Wywołanie MQPUT ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje kolejkę, do której dodawany jest komunikat, lub temat, do którego komunikat jest publikowany. Wartość *HOBJ* została zwrócona przez poprzednie wywołanie MQOPEN, które określiło opcję OOOUT.

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wysyłanego komunikatu i otrzymuje informacje na temat komunikatu po zakończeniu żądania umieszczenia. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, dane komunikatu można poprzężyć strukturą MQMDE w celu określenia wartości dla pól istniejących w deskrypcyjnie MQMD w wersji version-2, ale nie w wersji version-1. Pole *MDFMT* w strukturze MQMD musi być ustawione na FMMDE, aby wskazać, że jest obecna MQMDE. Więcej szczegółów na ten temat zawiera sekcja [“MQMDE-Rozszerzenie deskryptora komunikatu”](#) na stronie 1170.

PMO (MQPMO)-wejście/wyjście

Opcje sterujące działaniem MQPUT.

Szczegółowe informacje na ten temat zawiera sekcja [“MQPMO-Put-message, opcje”](#) na stronie 1191.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość komunikatu w produkcie *BUFFER*.

Wartość zero jest poprawna i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit dla *BUFLEN* zależy od różnych czynników:

- Jeśli kolejka docelowa jest kolejką współużytkowaną, górny limit wynosi 63 kB (64 512 bajtów).
- Jeśli miejsce docelowe jest kolejką lokalną lub jest tłumaczone na kolejkę lokalną (ale nie jest kolejką współużytkowaną), górną granicę zależy od tego, czy:
 - Lokalny menedżer kolejek obsługuje segmentację.
 - Aplikacja wysyłająca określa flagę, która umożliwia menedżerowi kolejek segmentowanie komunikatu. Ta opcja jest typu MFSEGA i może być określona albo w MQMD w wersji version-2, albo w produkcie MQMDE używanym z produktem MQMD w wersji version-1.

Jeśli oba te warunki zostaną spełnione, *BUFLEN* nie może przekroczyć 999 999 999 minus wartość pola *MDOFF* w deskrypcyjnie MQMD. Najdłuższy komunikat logiczny, który może zostać umieszczony, wynosi 999 999 999 bajtów (gdy *MDOFF* jest zerem). Jednak ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w których aplikacja jest uruchomiona, może spowodować obniżenie limitu.

Jeśli jeden lub oba z wcześniej opisanych warunków nie są spełnione, program *BUFLEN* nie może przekroczyć mniejszej wartości atrybutu *MaxMsgLength* kolejki i atrybutu *MaxMsgLength* menedżera kolejek.

- Jeśli miejsce docelowe jest kolejką zdalną lub jest tłumaczone na kolejkę zdalną, mają zastosowanie warunki dla kolejek lokalnych, *ale w każdym menedżerze kolejek, przez który musi przejść komunikat, aby dotrzeć do kolejki docelowej*; w szczególności:
 1. Lokalna kolejka transmisji używana do tymczasowego przechowywania komunikatu w lokalnym menedżerze kolejek
 2. Pośrednie kolejki transmisji (jeśli istnieją) używane do przechowywania komunikatu w menedżerach kolejek na trasie między lokalnymi i docelowymi menedżerami kolejek
 3. Kolejka docelowa w docelowym menedżerze kolejek

Najdłuższy komunikat, który może zostać umieszczony, jest zarządzany przez najbardziej restrykcyjne dla tych kolejek i menedżerów kolejek.

Gdy komunikat znajduje się w kolejce transmisji, dodatkowe informacje znajdują się wraz z danymi komunikatu, co zmniejsza ilość danych aplikacji, które mogą być przenoszone. W takiej sytuacji zaleca się odejmowanie bajtów LNMHD od wartości *MaxMsgLength* kolejek transmisji podczas określania limitu dla produktu *BUFLEN*.

Uwaga: Tylko niepowodzenie zgodności z warunkiem 1 może być diagnozowane synchronicznie (z kodem przyczyny RC2030 lub RC2031), gdy komunikat jest umieszczany. Jeśli warunki 2 lub 3 nie są spełnione, komunikat zostanie przekierowany do kolejki niedostarczonych komunikatów (niedostarczonych komunikatów), albo w pośrednim menedżerze kolejek, albo w docelowym menedżerze kolejek. Jeśli tak się stanie, zostanie wygenerowany komunikat raportu, o ile został on zażądany przez nadawcę.

BUFFER (1-bajtowy łańcuch bitowy x BUFLen)-wejście

Dane komunikatu.

Jest to bufor zawierający dane aplikacji, które mają zostać wysłane. Bufor powinien być wyrównany na granicy odpowiedniej do charakteru danych w komunikacie. 4-bajtowe wyrównanie powinno być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajтового.

Jeśli program *BUFFER* zawiera dane znakowe, dane liczbowe lub obie te wartości, pola *MDCSI* i *MDENC* w parametrze *MSGDSC* powinny być ustawione na wartości odpowiednie dla danych. To umożliwi odbiornikowi komunikatu przekształcenie danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* oraz w kodowaniu lokalnego menedżera kolejek nadawanego przez ENNAT.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2104

(2104, X'838 ') Opcja raportu w deskryptorze komunikatu nie została rozpoznana.

RC2136

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

Jeśli *CMPCOD* to CCFAIL:

RC2004

(2004, X'7D4') Parametr buforu nie jest poprawny.

RC2005

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2013

(2013, X'7DD') Czas utraty ważności nie jest poprawny.

RC2014

(2014, X'7DE') Kod sprzężenia zwrotnego jest niepoprawny.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2024

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

RC2026

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

RC2027

(2027, X'7EB') Brak odpowiedzi na kolejkę odpowiedzi.

RC2029

(2029, X'7ED') Typ komunikatu w deskrytorze komunikatu nie jest poprawny.

RC2030

(2030, X'7EE') Długość komunikatu jest większa niż maksymalna dla kolejki.

RC2031

(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2039

(2039, X'7F7') Kolejka nie jest otwarta dla danych wyjściowych.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2047

(2047, X'7FF') Trwałość nie jest poprawna.

RC2048

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

RC2050

(2050, X'802 ') Priorytet komunikatu nie jest poprawny.

RC2051

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2053

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2056

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2061

(2061, X'80D') Opcje raportu w deskrytorze komunikatu nie są poprawne.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2072

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2093

(2093, X'82D') Kolejka nie jest otwarta dla przekazywania wszystkich kontekstów.

RC2094

(2094, X'82E') Kolejka nie jest otwarta dla przekazywania kontekstu tożsamości.

RC2095

(2095, X'82F') Kolejka nie jest otwarta dla ustawiania całego kontekstu.

RC2096

(2096, X'830 ') Kolejka nie jest otwarta dla ustawiania kontekstu tożsamości.

RC2097

(2097, X'831 ') Uchwyt kolejki, o którym mowa, nie zapisuje kontekstu.

RC2098

(2098, X'832 ') Kontekst niedostępny dla uchwytu kolejki, o którym mowa.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2135

(2135, X'857 ') Struktura nagłówka dystrybucji nie jest poprawna.

RC2136

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

RC2137

(2137, X'859 ') Obiekt nie został otwarty pomyślnie.

RC2149

(2149, X'865 ') Konstrukcje PCF nie są poprawne.

RC2154

(2154, X'86A') Liczba obecnie niepoprawnych rekordów.

RC2156

(2156, X'86C') Rekordy odpowiedzi nie są poprawne.

RC2158

(2158, X'86E') flagi zapisu komunikatów nie są poprawne.

RC2159

(2159, X'86F') rekordów umieszczania komunikatów nie jest poprawna.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2173

(2173, X'87D') Struktura opcji put-message nie jest poprawna.

RC2185

(2185, X'889 ') Niespójna specyfikacja trwałości.

RC2188

(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

RC2189

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2241

(2241, X'8C1') Grupa komunikatów nie została zakończona.

RC2242

(2242, X'8C2') Komunikat logiczny nie został zakończony.

RC2245

(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2248

(2248, X'8C8') Rozszerzenie deskryptora komunikatu nie jest poprawne.

RC2249

(2249, X'8C9') Opcje komunikatu nie są poprawne.

RC2250

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

RC2251

(2251, X'8CB') Przesunięcie segmentu komunikatu nie jest poprawne.

RC2252

(2252, X'8CC') Oryginalna długość nie jest poprawna.

RC2253

(2253, X'8CD') Długość danych w segmencie komunikatów wynosi zero.

RC2255

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

RC2257

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

RC2258

(2258, X'8D2') Identyfikator grupy nie jest poprawny.

RC2266

(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

RC2269

(2269, X'8DD') Błąd zasobu klastra.

RC2270

(2270, X'8DE') Nie są dostępne żadne kolejki docelowe.

RC2420

(2420) Wywołanie MQPUT zostało wysłane, ale dane komunikatu zawierają strukturę MQEPH, która nie jest poprawna.

RC2479

(2479, X'9AF') Publikacja nie może być zachowana.

RC2480

(2480, X'9B0') Zmieniono typ docelowy: kolejka aliasowa odwołuje się do kolejki, ale teraz odnosi się do tematu.

RC2502

(2502, X'9C6') Publikacja nie powiodła się, a publikacja nie została dostarczona do żadnych subskrybentów

RC2551

(2551, X'9F7') Określony łańcuch wyboru nie jest dostępny.

RC2554

(2554, X'9FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

Deklaracja RPG

C*..1.....2.....3.....4.....5.....6.....7..

C	CALLP	MQPUT(HCONN : HOBJ : MSGDSC : PMO :
C		BUFLen : BUFFER : CMPCOD :
C		REASON)

Definicja prototypu dla wywołania to:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQPUT          PR          EXTPROC('MQPUT')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQPUT
D PMO          200A
D* Length of the message in Buffer
D BUFLen          10I 0 VALUE
D* Message data
D BUFFER          *   VALUE
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0

```

MQPUT1 -Umieść jeden komunikat

Wywołanie MQPUT1 umieszcza jeden komunikat w kolejce lub na liście dystrybucyjnej albo w temacie. Kolejka, lista dystrybucyjna lub temat nie muszą być otwarte.

- [“Składnia” na stronie 1368](#)
- [“Użycie notatek” na stronie 1368](#)
- [“Parametry” na stronie 1369](#)
- [“Deklaracja RPG” na stronie 1374](#)

Składnia

MQPUT1 (HCONN, OBJDSC, MSGDSC, PMO, BUFLen, BUFFER, CMPCOD, REASON)

Użycie notatek

1. Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. Wywołanie w celu użycia zależy od okoliczności:
 - Wywołanie MQPUT powinno być używane, gdy wiele komunikatów ma być umieszczonych w *tej samej* kolejce.
 Wywołanie MQOPEN z opcją OOOUT jest wysyłane jako pierwsze, po którym następuje jedna lub większa liczba żądań MQPUT w celu dodania komunikatów do kolejki. W końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1.
 - Wywołanie MQPUT1 powinno być używane, gdy tylko *jeden* komunikat ma zostać umieszczony w kolejce.
 Wywołanie to enkapsuluje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wysłane.
2. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez korzystania z grup komunikatów, kolejność tych komunikatów jest zachowywana, jeśli spełnione są określone warunki. Jednak w większości środowisk wywołanie MQPUT1 nie spełnia tych warunków, a więc nie zachowuje kolejności komunikatów. Zamiast tego w tych środowiskach należy użyć wywołania MQPUT. Szczegółowe informacje można znaleźć w uwagach dotyczących użycia w opisie wywołania MQPUT.

3. Wywołania MQPUT1 mogą być używane do umieszczania komunikatów w listach dystrybucyjnych. Ogólne informacje na ten temat można znaleźć w uwagach dotyczących użycia dla wywołań MQOPEN i MQPUT.

W przypadku korzystania z wywołania MQPUT1 występują następujące różnice:

- a. Jeśli aplikacja udostępniła rekordy odpowiedzi MQRR, muszą one być udostępniane przy użyciu struktury MQOD. Nie można ich używać przy użyciu struktury MQPMO.
- b. Kod przyczyny RC2137 nigdy nie jest zwracany przez MQPUT1 w rekordach odpowiedzi; jeśli kolejka nie zostanie otwarta, rekord odpowiedzi dla tej kolejki zawiera rzeczywisty kod przyczyny wynikający z operacji otwarcia.

Jeśli operacja otwarcia dla kolejki powiedzie się z kodem zakończenia CCWARN, kod zakończenia i kod przyczyny w rekordzie odpowiedzi dla tej kolejki są zastępowane przez kody zakończenia i przyczyny wynikające z operacji put.

Podobnie jak w przypadku wywołań MQOPEN i MQPUT, menedżer kolejek ustawia rekordy odpowiedzi (jeśli jest dostępne) tylko wtedy, gdy wynik wywołania nie jest taki sam dla wszystkich kolejek na liście dystrybucyjnej; jest to oznaczane przy użyciu wywołania kończonego z kodem przyczyny RC2136.

4. Jeśli wywołanie MQPUT1 jest używane do umieszczenia komunikatu w kolejce klastra, wywołanie zachowuje się tak, jakby w wywołaniu MQOPEN podano OOBNDN.
5. Jeśli na początku danych komunikatu aplikacji zostanie umieszczony komunikat zawierający jedną lub większą liczbę struktur nagłówka produktu IBM MQ, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka w celu sprawdzenia, czy są one poprawne. Więcej informacji na ten temat można znaleźć w uwagach dotyczących użycia wywołania MQPUT.
6. Jeśli wystąpi więcej niż jedna z sytuacji ostrzegawczych (patrz parametr *CMPCOD*), zwrócony kod przyczyny ma wartość *pierwsza* na następującej liście, która ma zastosowanie:
 - a. RC2136
 - b. RC2242
 - c. RC2241
 - d. RC2049 lub RC2104
7. Parametr *BUFFER* przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch; ograniczenie to ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr powinien zostać zadeklarowany jako struktura, albo jako pole w zbiorze fizycznym. Spowoduje to zwiększenie maksymalnej długości do około 32 kB.

Parametry

Wywołanie MQPUT1 ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

OBJDSC (MQOD)-wejście/wyjście

Deskryptor obiektu.

Jest to struktura identyfikująca kolejkę, do której dodawany jest komunikat. Szczegółowe informacje na ten temat zawiera sekcja [“MQOD-deskryptor obiektu”](#) na stronie 1176.

Użytkownik musi mieć uprawnienia do otwarcia kolejki dla danych wyjściowych. Kolejka modelowa **nie** musi być kolejką modelową.

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wysyłanego komunikatu i otrzymuje informację zwrotną po zakończeniu żądania umieszczenia. Szczegółowe informacje na ten temat zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Jeśli aplikacja udostępnia deskryptor MQMD w wersji version-1, dane komunikatu można poprzeżyć strukturą MQMDE w celu określenia wartości dla pól istniejących w deskrypcyjnie MQMD w wersji version-2, ale nie w wersji version-1. Pole *MDFMT* w strukturze MQMD musi być ustawione na FMMDE, aby wskazać, że jest obecna MQMDE. Więcej szczegółów na ten temat zawiera sekcja [“MQMDE-Rozszerzenie deskryptora komunikatu”](#) na stronie 1170.

PMO (MQPMO)-wejście/wyjście

Opcje sterujące działaniem komendy MQPUT1.

Szczegółowe informacje na ten temat zawiera sekcja [“MQPMO-Put-message, opcje”](#) na stronie 1191.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość komunikatu w produkcie *BUFFER*.

Wartość zero jest poprawna i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit zależy od różnych czynników. Więcej szczegółów można znaleźć w opisie parametru *BUFLEN* w wywołaniu MQPUT.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście

Dane komunikatu.

Jest to bufor zawierający dane komunikatu aplikacji, które mają zostać wysłane. Bufor powinien być wyrównany na granicy odpowiedniej do charakteru danych w komunikacie. 4-bajtowe wyrównanie powinno być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówka IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład: komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli program *BUFFER* zawiera dane znakowe, dane liczbowe lub obie te wartości, pola *MDCSI* i *MDENC* w parametrze *MSGDSC* powinny być ustawione na wartości odpowiednie dla danych. To umożliwi odbiornikowi komunikatu przekształcenie danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT1 muszą znajdować się w zestawie znaków podanym w atrybucie menedżera kolejek produktu *CodedCharSetId* i kodowaniu lokalnego menedżera kolejek podanego przez ENNAT.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2104

(2104, X'838 ') Opcja raportu w deskrytorze komunikatu nie została rozpoznana.

RC2136

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

RC2049

(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

RC2241

(2241, X'8C1') Grupa komunikatów nie została zakończona.

RC2242

(2242, X'8C2') Komunikat logiczny nie został zakończony.

Jeśli *CMPCOD* to CCFAIL:

RC2001

(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

RC2004

(2004, X'7D4') Parametr buforu nie jest poprawny.

RC2005

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2013

(2013, X'7DD') Czas utraty ważności nie jest poprawny.

RC2014

(2014, X'7DE') Kod sprzężenia zwrotnego jest niepoprawny.

RC2017

(2017, X'7E1') Nie ma więcej dostępnych uchwytów.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2024

(2024, X'7E8') Nie można obsłużyć więcej komunikatów w bieżącej jednostce pracy.

RC2026

(2026, X'7EA') deskryptor komunikatu nie jest poprawny.

RC2027

(2027, X'7EB') Brak odpowiedzi na kolejkę odpowiedzi.

RC2029

(2029, X'7ED') Typ komunikatu w deskrytorze komunikatu nie jest poprawny.

RC2030

(2030, X'7EE') Długość komunikatu jest większa niż maksymalna dla kolejki.

RC2031

(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2035

(2035, X'7F3') Brak uprawnień do dostępu.

RC2042

(2042, X'7FA') Obiekt jest już otwarty z opcjami powodujących konflikt.

RC2043

(2043, X'7FB') Typ obiektu nie jest poprawny.

RC2044

(2044, X'7FC') Struktura deskryptora obiektu nie jest poprawna.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2047

(2047, X'7FF') Trwałość nie jest poprawna.

RC2048

(2048, X'800 ') Kolejka nie obsługuje trwałych komunikatów.

RC2050

(2050, X'802 ') Priorytet komunikatu nie jest poprawny.

RC2051

(2051, X'803 ') Wywołania umieszczenia zablokowano dla kolejki.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2053

(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2056

(2056, X'808 ') Brak dostępnego miejsca na dysku dla kolejki.

RC2057

(2057, X'809 ') Typ kolejki nie jest poprawny.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2061

(2061, X'80D') Opcje raportu w deskrytorze komunikatu nie są poprawne.

RC2063

(2063, X'80F') Wystąpił błąd zabezpieczeń.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2072

(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2082

(2082, X'822 ') Nieznana kolejka podstawowa aliasu.

RC2085

(2085, X'825 ') Nieznana nazwa obiektu.

RC2086

(2086, X'826 ') Nieznany menedżer kolejek obiektów.

RC2087

(2087, X'827 ') Nieznany zdalny menedżer kolejek.

RC2091

(2091, X'82B') Kolejka transmisji nie jest lokalna.

RC2092

(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

RC2097

(2097, X'831 ') Uchwyt kolejki, o którym mowa, nie zapisuje kontekstu.

RC2098

(2098, X'832 ') Kontekst niedostępny dla uchwytu kolejki, o którym mowa.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2135

(2135, X'857 ') Struktura nagłówka dystrybucji nie jest poprawna.

RC2136

(2136, X'858 ') Zwrócenie wielu kodów przyczyny.

RC2149

(2149, X'865 ') Konstrukcje PCF nie są poprawne.

RC2154

(2154, X'86A') Liczba obecnie niepoprawnych rekordów.

RC2155

(2155, X'86B') Rekordy obiektów nie są poprawne.

RC2156

(2156, X'86C') Rekordy odpowiedzi nie są poprawne.

RC2158

(2158, X'86E') flagi zapisu komunikatów nie są poprawne.

RC2159

(2159, X'86F') rekordów umieszczania komunikatów nie jest poprawna.

RC2161

(2161, X'871 ') Menedżer kolejek jest wygaszany.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2173

(2173, X'87D') Struktura opcji put-message nie jest poprawna.

RC2184

(2184, X'888 ') Nazwa zdalnej kolejki nie jest poprawna.

RC2188

(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

RC2189

(2189, X'88D') Rezolucja nazwy klastra nie powiodła się.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2196

(2196, X'894 ') Nieznana kolejka transmisji.

RC2197

(2197, X'895 ') Nieznana domyślna kolejka transmisji.

RC2198

(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

RC2199

(2199, X'897 ') Domyślny błąd wykorzystania kolejki transmisji.

RC2258

(2258, X'8D2') Identyfikator grupy nie jest poprawny.

RC2248

(2248, X'8C8') Rozszerzenie deskryptora komunikatu nie jest poprawne.

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2249

(2249, X'8C9') Opcje komunikatu nie są poprawne.

RC2250

(2250, X'8CA') Numer kolejny komunikatu nie jest poprawny.

RC2251

(2251, X'8CB') Przesunięcie segmentu komunikatu nie jest poprawne.

RC2252

(2252, X'8CC') Oryginalna długość nie jest poprawna.

RC2253

(2253, X'8CD') Długość danych w segmencie komunikatów wynosi zero.

RC2255

(2255, X'8CF') Nie jest dostępna jednostka pracy dla menedżera kolejek, która ma być używana.

RC2257

(2257, X'8D1') Podano niewłaściwą wersję deskryptora MQMD.

RC2266

(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

RC2269

(2269, X'8DD') Błąd zasobu klastra.

RC2270

(2270, X'8DE') Nie są dostępne żadne kolejki docelowe.

RC2420

(2420) Wywołanie MQPUT1 zostało wysłane, ale dane komunikatu zawierają strukturę MQEPH, która nie jest poprawna.

RC2551

(2551, X'9F7') Określony tańcuch wyboru nie jest dostępny.

RC2554

(2554, X'9FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      MQPUT1(HCONN : OBJDSC : MSGDSC :
C                      PMO : BUFLN : BUFFER :
C                      CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQPUT1      PR          EXTPROC('MQPUT1')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object descriptor
D OBJDSC          468A
D* Message descriptor
D MSGDSC          364A
D* Options that control the action of MQPUT1
D PMO            200A
D* Length of the message in BUFFER
D BUFLN          10I 0 VALUE
D* Message data
D BUFFER          *   VALUE
D* Completion code
D CMPCOD          10I 0

```

MQSET-ustawienie atrybutów obiektu

Wywołanie MQSET służy do zmiany atrybutów obiektu reprezentowanego przez uchwyt. Obiekt musi być kolejką.

- [“Składnia” na stronie 1375](#)
- [“Użycie notatek” na stronie 1375](#)
- [“Parametry” na stronie 1375](#)
- [“Deklaracja RPG” na stronie 1379](#)

Składnia

MQSET (*HCONN*, *HOBJ*, *SELCNT*, *SELS*, *IACNT*, *INTATR*, *CALEN*, *CHRATR*, *CMPCOD*, *REASON*)

Użycie notatek

1. Za pomocą tego wywołania aplikacja może określić tablicę atrybutów całkowitoliczbowych lub kolekcję łańcuchów atrybutów znakowych. Jeśli nie wystąpią żadne błędy, podane atrybuty są ustawiane jednocześnie. W przypadku wystąpienia błędu (na przykład, jeśli selektor nie jest poprawny lub podjęto próbę ustawienia atrybutu na niepoprawną wartość), wywołanie nie powiedzie się i nie zostaną ustawione żadne atrybuty.
2. Wartości atrybutów można określić za pomocą wywołania MQINQ ; szczegółowe informacje na ten temat zawiera sekcja [“MQINQ-zapytanie o atrybuty obiektu” na stronie 1328](#).
Uwaga: Nie wszystkie atrybuty z wartościami, które mogą zostać zapytane przy użyciu wywołania MQINQ, mogą mieć zmienione wartości przy użyciu wywołania MQSET. Na przykład w przypadku tego wywołania nie można ustawić atrybutów procesu-obiektu lub menedżera kolejek.
3. Zmiany atrybutów są zachowywane po restartach menedżera kolejek (inne niż zmiany w tymczasowych kolejkach dynamicznych, które nie są restartowane restartami menedżera kolejek).
4. Nie można zmieniać atrybutów kolejki modelowej przy użyciu wywołania MQSET. Jeśli jednak kolejka modelowa zostanie otwarta za pomocą wywołania MQOPEN z opcją MQOO_SET, można użyć wywołania MQSET w celu ustawienia atrybutów dynamicznej kolejki lokalnej utworzonej przy użyciu wywołania MQOPEN.
5. Jeśli ustawiony obiekt jest kolejką klastra, musi istnieć lokalna instancja kolejki klastra, aby możliwe było pomyślne wykonanie tej kolejki.

 Więcej informacji na temat atrybutów obiektów zawiera sekcja:

- [“Atrybuty dla kolejek” na stronie 1395](#)
- [“Atrybuty dla list nazw” na stronie 1424](#)
- [“Atrybuty definicji procesów” na stronie 1425](#)
- [“Atrybuty dla menedżera kolejek” na stronie 1427](#)

Parametry

Wywołanie MQSET ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla HCONN :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt kolejki z atrybutami, które mają zostać ustawione. Uchwyt został zwrócony przez poprzednie wywołanie MQOPEN, które określiło opcję OOSSET.

SELCNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba selektorów.

Jest to liczba selektorów, które są dostarczane w macierzy SELS . Jest to liczba atrybutów, które mają zostać ustawione. Wartość zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

SELS (10-cyfrowa liczba całkowita ze znakiem x SELCNT)-dane wejściowe

Tablica selektorów atrybutów.

Jest to tablica selektorów atrybutów SELCNT ; każdy selektor identyfikuje atrybut (liczba całkowita lub znak) z wartością, która ma zostać ustawiona.

Każdy selektor musi być poprawny dla typu kolejki reprezentowanej przez produkt HOBJ . Dozwolone są tylko niektóre wartości IA* i CA* . Wartości te są wymienione w dalszej części tej sekcji.

Selektory mogą być określane w dowolnej kolejności. Wartości atrybutów, które odpowiadają selektorom atrybutów całkowitych (IA* selektory), muszą być określone w INTATR w tej samej kolejności, w jakiej te selektory występują w produkcie SELS . Wartości atrybutów, które odpowiadają selektorom atrybutów znakowych (CA* selektory), muszą być określone w CHRATR w tej samej kolejności, w jakiej występują te selektory. Selektory IA* można przeplatać się z selektorami CA* ; ważne jest tylko to, że kolejność względna w każdym typie jest istotna.

Nie jest błędem określenie tego samego selektora więcej niż raz; jeśli jest to zrobione, ostatnią wartością określoną dla konkretnego selektora jest ta, która staje się skuteczna.

Uwaga:

1. Selektory atrybutów całkowitych i atrybutów znakowych są przydzielane w dwóch różnych zakresach; selektory IA* znajdują się w zakresie IAFRST przez IALAST, a selektory CA* w zakresie CAFRST poprzez CALAST.

Dla każdego zakresu wartości stałe IALSTU i CALSTU definiują najwyższą wartość, którą menedżer kolejek zaakceptuje.

2. Jeśli wszystkie selektory IA* występują jako pierwsze, te same numery elementów mogą być używane do adresowania odpowiednich elementów w macierzach SELS i INTATR .

Atrybuty, które można ustawić, są wymienione w poniższej tabeli. Przy użyciu tego wywołania nie można ustawić żadnych innych atrybutów. W przypadku selektorów atrybutów CA* stała, która definiuje długość w bajtach łańcucha, który jest wymagany w produkcie CHRATR , jest podana w nawiasach.

Tabela 175. Selektory atrybutów MQSET dla kolejek		
Selektor	Opis	Uwaga
CATRGD	Dane wyzwalacza (LNTRGD).	<u>"2" na stronie 1377</u>

Tabela 175. Selektory atrybutów MQSET dla kolejek (kontynuacja)		
Selektor	Opis	Uwaga
IADIST	Obsługa listy dystrybucyjnej.	“1” na stronie 1377
IAIGET	Określa, czy operacje pobierania są dozwolone.	
IAIPUT	Określa, czy operacje put są dozwolone.	
IATRGC	Sterowanie wyzwalaczem.	“2” na stronie 1377
IATRGD	Wyzwalacz uruchamiany wypełnieniem.	“2” na stronie 1377
IATRGP	Priorytet komunikatu progowego dla wyzwalaczy.	“2” na stronie 1377
IATRGT	Typ wyzwalacza.	“2” na stronie 1377

Uwagi:

1. Obsługiwane tylko w klientach AIX, HP-UX, IBM i, Solaris, Windows oraz IBM MQ połączonych z tymi systemami.
2. Nieobsługiwane w systemie VSE/ESA.

IACNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba atrybutów całkowitych.

Jest to liczba elementów w tablicy *INTATR* i musi być ona co najmniej liczba selektorów IA* w parametrze *SELS*. Wartość zero jest poprawną wartością, jeśli nie istnieje żadna wartość.

INTATR (10-cyfrowa liczba całkowita ze znakiem x IACNT)-dane wejściowe

Tablica atrybutów całkowitoliczbowych.

Jest to tablica wartości atrybutu całkowitoliczbowego *IACNT*. Te wartości atrybutów muszą być w tej samej kolejności, w jakiej znajdują się selektory IA* w tablicy *SELS*.

CALEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość buforu atrybutów znakowych.

Jest to długość w bajtach parametru *CHRA TR*, która musi być co najmniej równa sumie długości atrybutów znakowych określonych w tablicy *SELS*. Wartość zero jest poprawną wartością, jeśli nie ma selektorów CA* w produkcie *SELS*.

CHRA TR (1-bajtowy łańcuch znaków x CALEN)-wejście

Atrybuty znaków.

Jest to bufor zawierający wartości atrybutów znakowych, które są konkatelowane. Długość buforu jest nadawana przez parametr *CALEN*.

Atrybuty znaków muszą być określone w tej samej kolejności, w jakiej znajdują się selektory CA* w tablicy *SELS*. Długość każdego atrybutu znakowego jest stała (patrz *SELS*). Jeśli wartość, która ma być ustawiona dla atrybutu, zawiera mniej znaków niepustych niż zdefiniowana długość atrybutu, wartość w polu *CHRA*TR musi być dopełniona z prawej strony znakami odstępu, aby wartość atrybutu była zgodna ze zdefiniowaną długością atrybutu.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFail:

RC2219

(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2006

(2006, X'7D6') Długość atrybutów znakowych nie jest poprawna.

RC2007

(2007, X'7D7') Łańcuch atrybutów znakowych nie jest poprawny.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2019

(2019, X'7E3') Uchwyt obiektu nie jest poprawny.

RC2020

(2020, X'7E4') Wartość atrybutu inhibit-get lub inhibit-put nie jest poprawna.

RC2021

(2021, X'7E5') Liczba atrybutów całkowitych nie jest poprawna.

RC2023

(2023, X'7E7') Tablica atrybutów Integer nie jest poprawna.

RC2040

(2040, X'7F8') Kolejka nie jest otwarta do ustawienia.

RC2041

(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101

(2101, X'835 ') Obiekt jest uszkodzony.

RC2052

(2052, X'804 ') Kolejka została usunięta.

RC2058

(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nie jest znana.

RC2059

(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162

(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2065

(2065, X'811 ') Count of selectors not valid.

RC2067

(2067, X'813 ') Selektor atrybutu nie jest poprawny.

RC2066

(2066, X'812 ') Count of selectors too large.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2075

(2075, X'81B') Wartość dla atrybutu sterującego wyzwalacza jest niepoprawna.

RC2076

(2076, X'81C') Wartość atrybutu głębokości wyzwalacza nie jest poprawna.

RC2077

(2077, X'81D') Wartość atrybutu wyzwalacza-message-priority nie jest poprawna.

RC2078

(2078, X'81E') Wartość atrybutu wyzwalacza nie jest poprawna.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C              CALLP      MQSET(HCONN : HOBJ : SELCNT :
C                      SELS(1) : IACNT : INTATR(1) :
C                      CALEN : CHRATR : CMPCOD :
C                      REASON)

```

Definicja prototypu dla wywołania to:

```

D*..1.....2.....3.....4.....5.....6.....7..
DMQSET          PR          EXTPROC('MQSET')
D* Connection handle
D HCONN          10I 0 VALUE
D* Object handle
D HOBJ          10I 0 VALUE
D* Count of selectors
D SELCNT          10I 0 VALUE
D* Array of attribute selectors
D SELS          10I 0
D* Count of integer attributes
D IACNT          10I 0 VALUE
D* Array of integer attributes
D INTATR          10I 0
D* Length of character attributes buffer
D CALEN          10I 0 VALUE
D* Character attributes
D CHRATR          *   VALUE
D* Completion code
D CMPCOD          10I 0
D* Reason code qualifying CMPCOD
D REASON          10I 0

```

MQSETMP-ustawianie właściwości uchwytu komunikatu

Wywołanie MQSETMP ustawia lub modyfikuje właściwość uchwytu komunikatu.

- [“Składnia” na stronie 1380](#)
- [“Użycie notatek” na stronie 1380](#)
- [“Parametry” na stronie 1381](#)
- [“Deklaracja RPG” na stronie 1384](#)

Składnia

MQSETMP (*Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, CompCode, Reason*)

Użycie notatek

- Tego wywołania można użyć tylko wtedy, gdy menedżer kolejek sam koordynuje jednostkę pracy. Może to być:
 - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
 - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .
- Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja [“MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 1274](#).
- W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiednich wywołań zwrotnych zamiast MQBACK. Środowisko może również obsługiwać niejawne wycofania spowodowane przez nieprawidłowe zakończenie działania aplikacji.
 - W systemie z/OS należy użyć następujących wywołań:
 - Programy wsadowe (w tym wsadowe programy DL/I produktu IMS) mogą używać wywołania MQBACK, jeśli jednostka pracy ma wpływ tylko na zasoby IBM MQ . Jeśli jednak jednostka pracy ma wpływ na zasoby zarówno IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnionego przez usługę RRS (Recoverable Resource Service) produktu z/OS . Wywołanie SRRBACK powoduje wycofania zmian w zasobach należących do menedżerów zasobów, które zostały włączone dla koordynacji RRS.
 - Aplikacje produktu CICS muszą używać komendy EXEC CICS SYNCPOINT ROLLBACK do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
 - Aplikacje produktu IMS (inne niż wsadowe programy DL/I) muszą używać wywołań programu IMS , takich jak ROLB , do utworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż wsadowe programy DL/I).
 - W systemie IBM należy użyć tego wywołania dla lokalnych jednostek pracy koordynowanych przez menedżera kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem CMTSCOPE (*JOB) nie może zostać wydana dla zadania.
- Jeśli aplikacja kończy się z niezatwierdzoną zmianą w jednostce pracy, to dyspozycja tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy też nieprawidłowo. Więcej szczegółów można znaleźć w uwagach dotyczących użycia w produkcie [“MQDISC-rozłączenie menedżera kolejek” na stronie 1312](#) .
- Gdy aplikacja wstawi lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Informacje te są powiązane z uchwytami kolejki i obejmują takie elementy jak:
 - Wartości pól *GroupId, MsgSeqNumber, Offset i MsgFlags* w strukturze MQMD.
 - Określa, czy komunikat jest częścią jednostki pracy.

- W przypadku wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje trzy zestawy informacji o grupach i segmentach, jeden zestaw dla każdego z następujących elementów:

- Ostatnie pomyślne wywołanie MQPUT (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które przeglądało komunikat w kolejce (nie może to być część jednostki pracy).

Jeśli aplikacja wstawi lub pobiera komunikaty jako część jednostki pracy, a następnie aplikacja podejmie decyzję o wytworzeniu kopii zapasowej jednostki pracy, informacje o grupie i segmencie zostaną odtworzone do wartości, którą poprzednio:

- Informacje powiązane z wywołaniem MQPUT są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
- Informacje powiązane z wywołaniem MQGET są przywracane do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zakresem jednostki pracy, nie mają informacji o grupach i segmentach, które zostały odtworzone, jeśli zostanie utworzona kopia zapasowa jednostki pracy.

Odtwarzanie informacji o grupach i segmentach do jej poprzedniej wartości, gdy tworzona jest kopia zapasowa jednostki pracy, umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy, a także zrestartowanie w poprawnym punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się.

Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną kolejkę pamięci masowej. Jednak aplikacja musi zachować wystarczające informacje, aby móc restartować wprowadzanie lub pobieranie komunikatów w poprawnym punkcie, jeśli wystąpi awaria systemu.

Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu można znaleźć w sekcji PMLOGO opisanej w sekcji PMOPT (10-cyfrowa liczba podpisanych cyfr), a także w opcji GMLOGO opisanej w sekcji GMOPT (10-cyfrowa liczba całkowita ze znakiem).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

- Jednostka pracy ma ten sam zasięg co uchwyt połączenia. Wszystkie wywołania produktu IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wydane przy użyciu innego uchwytu połączenia (na przykład wywołania wydane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń znajdują się w sekcji HCONN (10-cyfrowa liczba całkowita ze znakiem)-wyjście .
- To wywołanie ma wpływ tylko na komunikaty, które zostały wprowadzone lub pobrane jako część bieżącej jednostki pracy.
- Długo działająca aplikacja, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapełnić kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć tę możliwość, administrator musi ustawić atrybut menedżera kolejek produktu *MaxUncommittedMsgs* na wartość, która jest na tyle niska, aby zapobiec zapełnieniu kolejek przez aplikacje w trybie runaway, ale na tyle duże, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQSETMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze *HMSG* .

Jeśli uchwyt komunikatu został utworzony przy użyciu komendy HCUNAS, należy ustawić poprawne połączenie w wątku ustawiające właściwość uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się z kodem przyczyny RC2009 .

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

To jest uchwyt komunikatu, który ma zostać zmodyfikowany. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

SETOPT (MQSMPO)-wejście

Sterowanie sposobem ustawiania właściwości komunikatu.

Ta struktura umożliwia aplikacjom określanie opcji sterujących sposobem ustawiania właściwości komunikatu. Struktura jest parametrem wejściowym w wywołaniu MQSETMP. Więcej informacji na ten temat zawiera sekcja [MQSMPO](#) .

PRNAME (MQCHARV)-dane wejściowe

Jest to nazwa właściwości do ustawienia.

Więcej informacji na temat korzystania z nazw właściwości zawiera sekcja [Nazwy właściwości i ograniczenia dotyczące nazw właściwości](#) .

PRPDSC (MQPD)-wejście/wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym:

- co się stanie, jeśli właściwość nie jest obsługiwana
- jaki kontekst komunikatu, do której należy właściwość
- Jakie komunikaty są kopiowane do postaci, w której jest ona kopiowana

Więcej informacji na temat tej struktury zawiera sekcja [MQPD](#) .

TYPE (10-cyfrowa liczba całkowita ze znakiem)-wejście

Typ danych dla ustawianej właściwości. Może to być jeden z następujących elementów:

TYPBOL

Wartość boolowska. *ValueLength* musi mieć wartość 4.

TYPBST

Łańcuch bajtów. *ValueLength* musi być równe zero lub większe.

TYPI8

8-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 1.

TYPI16

16-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 2.

TYPI32

32-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 4.

TYPI64

64-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 8.

TYPF32

32-bitowa liczba zmiennoprzecinkowa. *ValueLength* musi mieć wartość 4.

TYPF64

64-bitowa liczba zmiennopozycyjna. *ValueLength* musi mieć wartość 8.

TYPSTR

Łańcuch znaków. *ValueLength* musi mieć wartość zero lub większą, albo wartość specjalną VLNULL.

TYPNUL

Właściwość istnieje, ale ma wartość NULL. *ValueLength* musi mieć wartość zero.

VALLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość (w bajtach) wartości właściwości w parametrze *Wartość*.

Wartość zero jest poprawna tylko dla wartości NULL lub łańcuchów lub łańcuchów bajtów. Wartość zero wskazuje, że właściwość istnieje, ale nie zawiera żadnych znaków ani bajtów.

Jeśli parametr *Type* ma ustawiony parametr TYPSTR, wartość ta musi być większa lub równa zero lub musi być równa zero lub jest równa następującej wartości specjalnej:

VLNULL

Wartość jest ograniczona do pierwszej wartości null napotkanej w łańcuchu. Wartość NULL nie jest uwzględniana jako część łańcucha. Ta wartość jest niepoprawna, jeśli parametr TYPSTR nie jest również ustawiony.

Uwaga: znak o kodzie zero używany do zakończenia łańcucha, jeśli wartość VLNULL jest ustawiona, to wartość NULL jest wartością z zestawu znaków wartości.

VALUE (1-bajtowy łańcuch bitowy x VALLEN)-wejście

Wartość właściwości, która ma zostać ustawiona. Bufor musi być wyrównany na granicy odpowiedniej do charakteru danych w wartości.

W języku programowania C parametr ten jest zadeklarowany jako wskaźnik-do-void; adres dowolnego typu danych może być określony jako parametr.

Jeśli parametr *ValueLength* ma wartość zero, to nie jest przywołana wartość *Value*. W tym przypadku adres parametru przekazywany przez programy napisane w języku C lub System/390 assembler może mieć wartość NULL.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli parametr *CMPCOD* ma wartość CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CMPCOD* ma wartość CCWARN:

RC2421

(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli parametr *CMPCOD* ma wartość CCFail:

RC2204

(2204, X'089C') Adapter nie jest dostępny.

RC2130

(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157

(2157, X'86D') Identyfikatory ASID podstawowego i podstawowego różnią się.

RC2004

(2004, X'07D4') Parametr Wartość nie jest poprawny.

RC2005

(2005, X'07D5') Parametr długości wartości nie jest poprawny.

RC2219

(2219, X'08AB') Wywołanie MQI zostało wprowadzone przed zakończeniem poprzedniego wywołania.

RC2460

(2460, X'099C') Wskaźnik uchwytu komunikatu nie jest poprawny.

RC2499

(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046

(2046, X'07FE') Opcje nie są poprawne lub niespójne.

RC2482

(2482, X'09B2') Struktura deskryptora właściwości nie jest poprawna.

RC2442

(2442, X'098A') Niepoprawna nazwa właściwości.

RC2473

(2473, X'09A9') Niepoprawny typ danych właściwości.

RC2472

(2472, X'09A8') Napotkano błąd formatu liczb w danych wartości.

RC2463

(2463, X'099F') Ustawianie struktury opcji właściwości komunikatu nie jest poprawne.

RC2111

(2111, X'083F') Identyfikator kodowanego zestawu znaków nazwy właściwości nie jest poprawny.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej szczegółów na ten temat zawiera sekcja [“Kody powrotu dla IBM i \(ILE RPG\)”](#) na stronie 1455.

Deklaracja RPG

```

C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP      MQSETMP(HCONN : HMSG : SETOPT :
                                           PRNAME : PRPDSC :
                                           TYPE : VALLEN : VALUE :
                                           CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

DMQSETMP      PR      EXTPROC('MQSETMP')
D* Connection handle
D HCONN              10I 0 VALUE
D* Message handle
D HMSG              10I 0 VALUE
D* Options that control the action of MQSETMP
D SETOPT            20A
D* Property name
D PRNAME            32A
D* Property descriptor
D PRPDSC            24A
D* Property data type
D TYPE              10I 0 VALUE
D* Length of the Value area
D VALLEN            10I 0 VALUE
D* Property value

```

D VALUE	*	VALUE
D* Completion code		
D CMPCOD	10I	0
D* Reason code qualifying CompCode		
D REASON	10I	0

MQSTAT-pobieranie informacji o statusie

Użyj wywołania MQSTAT, aby pobrać informacje o statusie. Typ zwracanych informacji o statusie jest określany na podstawie wartości STYPE określonej w wywołaniu.

- [“Składnia” na stronie 1385](#)
- [“Użycie notatek” na stronie 1385](#)
- [“Parametry” na stronie 1385](#)
- [“Deklaracja RPG” na stronie 1386](#)

Składnia

MQSTAT (*HCONN*, *STYPE*, *STAT*, *CMPCOD*, *REASON*)

Użycie notatek

1. Wywołanie MQSTAT, określając typ STATAPT, zwraca informacje o poprzednich asynchronicznych operacjach MQPUT i MQPUT1. Struktura MQSTAT przekazana w wywołaniu została zakończona z pierwszym zarejestrowanym asynchronicznym ostrzeżeniem lub informacją o błędzie dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia będą następowały po pierwszym, nie zmieniają one zwykle tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia CCWARN, to zamiast tego zwracana jest kolejna awaria z kodem zakończenia CCFAIL.
2. Jeśli od momentu nawiązania połączenia lub od ostatniego wywołania MQSTAT nie wystąpiły żadne błędy, zwracane są CMPCOD o wartości CCOK i PRZYCZYNA RCNONE.
3. Liczby wywołań asynchronicznych, które zostały przetworzone w ramach uchwytu połączenia, są zwracane przy użyciu trzech liczników: STSPSC, STSPWC i STSPFC. Liczniki te są zwiększane przez menedżer kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy zwrócić uwagę, że w celach księgowych lista dystrybucyjna jest liczona raz dla kolejki docelowej, a nie raz na listę dystrybucyjną).
4. Pomyślne wywołanie komendy MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczby.

Parametry

Wywołanie MQSTAT ma następujące parametry:

Hconn (MQHCONN)-dane wejściowe

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

STYPE (10-cyfrowa liczba całkowita ze znakiem)-wejście

Typ żądanych informacji o statusie. Jedyną poprawną wartością jest:

STATAPT

Zwraca informacje na temat poprzednich asynchronicznych operacji put.

STS (MQSTS)-wejście/wyjście

Struktura informacji o statusie. Szczegółowe informacje na ten temat zawiera sekcja [“MQSTS-struktura raportowania statusu” na stronie 1250](#).

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Zakończenie powiodło się.

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2374

(2374, X' 946 ') Wyjście interfejsu API nie powiodło się

RC2183

(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2219

(2219, X'8AB') Wywołanie MQI zostało wprowadzone przed ukończonym wcześniejszym wywołaniem.

RC2009

(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2203

(2203, X'89B') Połączenie jest zamykane.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2162

(2162, X'872 ') Zatrzymywanie menedżera kolejek

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2430

(2430, X'97E') Błąd typu MQSTAT.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2424

(2424, X' 978 ') Błąd struktury MQSTS

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2298

(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Szczegółowe informacje na temat tych kodów można znaleźć w:

- [Komunikaty i kody przyczyny](#)

Deklaracja RPG

```

C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C                                     CALLP      MQSTAT(HCONN : ETYPE : ERR :
C                                     CMPCOD : REASON)

```

Definicja prototypu dla wywołania to:

```

D.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7

```

DMQSTAT	PR	EXTPROC('MQSTAT')
D* Connection handle		
D HCONN		10I 0 VALUE
D* Status information type		
D STYPE		10I 0 VALUE
D* Status information		
D STATUS		296A
D* Completion code		
D CMPCOD		10I 0
D* Reason code qualifying CompCode		
D REASON		10I 0

MQSUB-Zarejestruj subskrypcję

Wywołanie MQSUB powoduje zarejestrowanie subskrypcji aplikacji w określonym temacie.

- [“Składnia” na stronie 1387](#)
- [“Użycie notatek” na stronie 1387](#)
- [“Parametry” na stronie 1389](#)
- [“Deklaracja RPG” na stronie 1393](#)

Składnia

MQSUB (*HCONN*, *SUBDSC*, *HOBJ*, *HSUB*, *CMPCOD*, *REASON*)

Użycie notatek

- Subskrypcja jest tworzona w temacie o nazwie określonej za pomocą skróconej nazwy predefiniowanego obiektu tematu, pełnej nazwy łańcucha tematu lub konkatenacji dwóch części, zgodnie z opisem w sekcji [“Korzystanie z łańcuchów tematów” na stronie 584](#).
- Menedżer kolejek wykonuje sprawdzenia zabezpieczeń po wywołaniu wywołania MQSUB w celu sprawdzenia, czy identyfikator użytkownika, pod którym aplikacja jest uruchomiona, ma odpowiedni poziom uprawnień, zanim dostęp jest dozwolony. Odpowiedni obiekt tematu znajduje się albo za pomocą krótkiej nazwy podanej w wywołaniu, albo najbliższego obiektu nazwy skróconej w hierarchii tematów, który zostanie znaleziony, jeśli zostanie podana długa nazwa. W tym obiekcie tematu wykonywane jest sprawdzenie uprawnień, aby upewnić się, że ustawiono uprawnienia do subskrybowania oraz w kolejce docelowej, aby upewnić się, że ustawione są uprawnienia dla danych wyjściowych. Jeśli używana jest opcja SDMAN, oznacza to, że w nazwie kolejki zarządzanej powiązanej z tym obiektem tematu jest wykonywane sprawdzenie uprawnień, a jeśli podana jest kolejka niezarządzana, oznacza to, że w kolejce reprezentowanej przez parametr *HOBJ* jest wykonywane sprawdzenie uprawnień.
- *HOBJ* zwrócona w wywołaniu MQSUB, gdy używana jest opcja SOMAN, można sprawdzić, aby znaleźć atrybuty, takie jak próg wycofania i Nadmierna nazwa kolejki wycofanych komunikatów. Można również zapytać o nazwę kolejki zarządzanej, ale nie należy próbować bezpośrednio otwierać tej kolejki.
- Subskrypcje można grupować, zezwalając na dostarczanie tylko jednej publikacji do grupy subskrypcji, nawet jeśli więcej niż jedna grupa jest zgodna z publikacją. Subskrypcje są grupowane za pomocą opcji SOGRP oraz w celu grupowania subskrypcji, które muszą:
 - korzystać z tej samej kolejki nazwanej (która nie korzysta z opcji SOMAN) w tym samym menedżerze kolejek-reprezentowanym przez parametr *HOBJ* w wywołaniu MQSUB
 - współużytkuj ten sam *SDCID*
 - być z tego samego *SDSL*

Atrybuty te definiują zestaw subskrypcji uważanych za znajdujące się w grupie, a także atrybuty, których nie można zmienić, jeśli subskrypcja została pogrupowana. Zmiana *SDSL* wyników w programie RC2512 oraz zmiana dowolnego z pozostałych (które można zmienić, jeśli subskrypcja nie jest zgrupowana) powoduje RC2515.
- Pola w tabeli MQSD są wykonywane po powrocie z wywołania MQSUB, który korzysta z opcji SORES. Zwracaną wartość MQSD można przekazać bezpośrednio do wywołania MQSUB, które korzysta z opcji

SOALT z dowolnymi zmianami, które należy wprowadzić w ramach subskrypcji zastosowanego w MQSD. Niektóre pola mają specjalne uwagi, które zostały odnotowane w tabeli.

Dane wyjściowe MQSD z tabeli MQSUB	
Nazwa pola w MQSD	Uwagi szczególne
Opcje dostępu lub tworzenia	Żadna z tych opcji nie jest ustawiona po powrocie z wywołania MQSUB. Jeśli później ponownie zostanie ponownie użyta wartość MQSD w wywołaniu MQSUB, konieczne jest jawne ustawienie tej opcji.
Opcje trwałości, Opcje docelowe, Opcje rejestracji i opcje wieloznaczne	Te opcje zostaną ustawione jako odpowiednie
Opcje publikacji	Opcje te zostaną określone jako odpowiednie, z wyjątkiem SONEWP, które ma zastosowanie wyłącznie do SOCRE.
Inne opcje	Te opcje nie zmieniają się po powrocie z wywołania MQSUB. Sterują one sposobem, w jaki wywołanie API jest wysyłane i nie jest zapisywane w subskrypcji. Muszą one być ustawione zgodnie z wymaganiami w przypadku kolejnych wywołań MQSUB ponownie korzystających z MQSD.
ObjectName	To pole wejściowe jest tylko niezmienione w przypadku zwrotu z wywołania MQSUB.
ObjectString	To pole wejściowe jest tylko niezmienione w przypadku zwrotu z wywołania MQSUB. W polu <i>SDRO</i> zwracana jest pełna nazwa tematu, jeśli został podany bufor.
Identyfikator AlternateUser i identyfikator AlternateSecurity	Te pola wejściowe są tylko niezmienione po powrocie z wywołania MQSUB. Sterują one sposobem, w jaki wywołanie API jest wysyłane i nie jest zapisywane w subskrypcji. Muszą one być ustawione zgodnie z wymaganiami w przypadku kolejnych wywołań MQSUB ponownie korzystających z wywołania MQSD.
SubExpiry	W przypadku powrotu z wywołania MQSUB przy użyciu opcji SORES to pole zostanie ustawione na pierwotne wygaśnięcie subskrypcji, a nie pozostały czas utraty ważności. Jeśli zmaterializowana tabela MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB za pomocą opcji SOALT, zostanie zresetowane wygaśnięcie subskrypcji w celu ponownego zliczania czasu.
SubName	To pole jest polem wejściowym w wywołaniu MQSUB i nie jest zmieniane na wyjściu.

Dane wyjściowe MQSD z tabeli MQSUB (kontynuacja)	
Nazwa pola w MQSD	Uwagi szczególne
Dane SubUseri SelectionString	Te pola o zmiennej długości zostaną zwrócone w wyniku wywołania MQSUB za pomocą opcji SORES, jeśli bufor jest udostępniony, a także dodatnia długość buforu w produkcie VCHRP. Jeśli żaden bufor nie zostanie podany, tylko długość zostanie zwrócona w polu VCHRL obiektu MQCHARV.If podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w udostępnionym buforze zwracane są tylko VCHRP bajtów. Jeśli zmaterializowana tabela MQSD zostanie później ponownie wykorzystana w wywołaniu MQSUB za pomocą opcji SOALT, a bufor nie zostanie podany, ale zostanie podana wartość inna niż zero VCHRL , to jeśli ta długość będzie zgodna z istniejącą długością pola, wówczas żadna zmiana nie zostanie wykonana w tym polu.
Identyfikator SubCorreli znacznik PubAccounting	Jeśli identyfikator SOSCID nie zostanie użyty, program SDCID zostanie wygenerowany przez menedżer kolejek. Jeśli SOSETI nie jest używane, SDACC zostanie wygenerowany przez menedżer kolejek. Te pola zostaną zwrócone w tabeli MQSD z wywołania MQSUB przy użyciu opcji SORES. Jeśli są one generowane przez menedżer kolejek, wygenerowana wartość zostanie zwrócona w wywołaniu MQSUB za pomocą opcji SOCRE lub SOALT.
PubPriority, SubLevel & PubApplIdentityData	Pola te zostaną zwrócone w tabeli MQSD.
Łańcuch ResObject	To pole wyjściowe zostanie zwrócone w MQSD tylko wtedy, gdy zostanie udostępniony bufor.

Parametry

Wywołanie MQSUB ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość HCONN została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla HCONN :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

SUBDSC (MQSD)-wejście/wyjście

Jest to struktura identyfikująca obiekt z użyciem, który jest rejestrowany przez aplikację. Więcej informacji na ten temat zawiera sekcja [“MQSD-deskryptor subskrypcji” na stronie 1231](#).

HOBj (10-cyfrowa liczba całkowita ze znakiem)-input/output

Ten uchwyt reprezentuje dostęp, który został utworzony w celu uzyskania komunikatów wysłanych do tej subskrypcji. Komunikaty te mogą być przechowywane w określonej kolejce lub menedżer kolejek może zostać poproszony o zarządzanie pamięcią masową bez konieczności wykonania określonej kolejki.

Uchwyt obiektu.

Jeśli określona kolejka ma być używana, musi być powiązana z subskrypcją w czasie tworzenia. Można to zrobić na dwa sposoby:

- Udostępniając ten uchwyt podczas wywoływania komendy MQSUB z opcją SDCRT. Jeśli ten uchwyt zostanie podany jako parametr wejściowy w wywołaniu, musi to być poprawny uchwyt obiektu zwrócony z poprzedniego wywołania MQOPEN kolejki przy użyciu co najmniej jednego z OOINP*, OOOOUT (jeśli na przykład kolejki zdalnej) lub opcji OOBROW. Jeśli tak nie jest, wywołanie kończy się niepowodzeniem z kodem RC2019. Nie może to być uchwyt obiektu do kolejki aliasowej, która jest tłumaczona na obiekt tematu. Jeśli tak, wywołanie kończy się niepowodzeniem z kodem RC2019.
- Używając komendy DEFINE SUB MQSC, i pod warunkiem, że komenda ta ma nazwę obiektu kolejki.

Jeśli menedżer kolejek ma zarządzać pamięcią masową komunikatów wysłanych do tej subskrypcji, należy wskazać to, kiedy subskrypcja jest tworzona, za pomocą opcji SOMAN i ustawiając wartość parametru na HONONE. Menedżer kolejek zwraca uchwyt jako parametr wyjściowy w wywołaniu, a zwracany uchwyt jest znany jako uchwyt zarządzany. Jeśli podano parametr HONONE, a parametr SOMAN nie jest określony, wywołanie kończy się niepowodzeniem z kodem RC2019.

Zarządzany uchwyt zwracany przez menedżera kolejek może być używany w wywołaniu MQGET lub MQCB, z opcjami przeglądania lub bez, w wywołaniu MQINQ lub w tabeli MQCLOSE. Nie można jej używać w przypadku operacji MQPUT, MQSET ani w późniejszym czasie operacji MQSUB; próba wykonania tego działania kończy się niepowodzeniem z kodem RC2039 dla MQPUT, RC2040 dla MQSET lub RC2038 dla MQSUB.

Jeśli opcja SORES w polu OPTS w strukturze MQSD jest używana do wznowienia tej subskrypcji, uchwyt może zostać zwrócony do aplikacji w tym parametrze, jeśli podano wartość HONONE. Można tego użyć, niezależnie od tego, czy subskrypcja jest używana, czy nie. Może to być przydatne dla subskrypcji utworzonych przy użyciu opcji DEFINE SUB, jeśli uchwyt ma być używany do kolejki subskrypcji zdefiniowanej w komendzie DEFINE SUB. W przypadku, gdy wznowiana jest administracyjna subskrypcja, kolejka jest otwierana z OOINPQ i OOBROW. Jeśli wymagane są inne opcje, aplikacja musi jawnie otworzyć kolejkę subskrypcji i udostępnić uchwyt obiektu w wywołaniu. Jeśli wystąpi problem z otwarciem kolejki, wywołanie zakończy się niepowodzeniem z kodem RC2522. Jeśli zostanie podana HOBj, musi to być odpowiednik HOBj w oryginalnym wywołaniu MQSUB. Oznacza to, że jeśli udostępniany jest uchwyt obiektu zwrócony z wywołania MQOPEN, uchwyt musi być w tej samej kolejce, co poprzednio używane lub wywołanie kończy się niepowodzeniem z kodem RC2019.

Jeśli ta subskrypcja jest zmieniana, za pomocą opcji SOALT w polu OPTS w strukturze MQSD można podać inną wartość HOBj. Wszystkie publikacje, które zostały dostarczone do kolejki wcześniej zidentyfikowanej za pomocą tego parametru, pozostają w tej kolejce i należy je pobrać w celu pobrania tych komunikatów, jeśli parametr HOBj reprezentuje teraz inną kolejkę.

Użycie tego parametru z różnymi opcjami subskrypcji zostało podsumowane w poniższej tabeli:

Opcje	Hobj	Opis
SOCRT + SOMAN	Ignorowane na wejściu	Tworzy subskrypcję za pomocą menedżera kolejek, który jest zarządzany przez pamięć masową komunikatów.

Opcje	Hobj	Opis
SOCRT	Poprawny uchwyt obiektu	Tworzy subskrypcję udostępniając określoną kolejkę jako miejsce docelowe dla komunikatów.
SORES	HONONE	Wznawia wcześniej utworzoną subskrypcję (zarządzaną lub nie), a menedżer kolejek zwraca uchwyt obiektu do użycia przez aplikację.
SORES	Poprawne, zgodne, uchwyt obiektu	Wznawia wcześniej utworzoną subskrypcję, która używa określonej kolejki jako miejsca docelowego dla komunikatów i korzysta z uchwytu obiektu z konkretnymi opcjami otwarcia.
SOALT + SOMAN	HONONE	Zmienia istniejącą subskrypcję, która wcześniej używała konkretnej kolejki, do tej pory zarządzaną.
SOALT	Poprawny uchwyt obiektu	Zmienia istniejącą subskrypcję w celu użycia określonej kolejki (z zarządzanego lub z innej konkretnej kolejki).

Niezależnie od tego, czy został on dostarczony, czy też został zwrócony, należy podać *HOBJ* w kolejnych wywołaniach MQGET, które muszą zostać wyświetlone.

Uchwyt *HOBJ* przestaje być poprawny, gdy na nim zostanie wywołane wywołanie MQCLOSE, lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostaje zakończona. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Informacje na temat uchwytu zasięgu można znaleźć w sekcji [HCONN](#). Operacja MQCLOSE uchwytu *HOBJ* nie ma wpływu na uchwyt *HSUB*.

HSUB (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Ten uchwyt reprezentuje subskrypcję, która została wykonana. Może być używany do dwóch kolejnych operacji:

- Można go użyć w kolejnym wywołaniu MQSUBRQ, aby zażądać wysłania publikacji, gdy opcja SOPUBR została użyta podczas dokonywania subskrypcji.
- Można go użyć podczas kolejnego wywołania MQCLOSE w celu usunięcia subskrypcji, która została wykonana. Uchwyt *HSUB* przestaje być poprawny, gdy zostanie wywołane wywołanie MQCLOSE lub gdy jednostka przetwarzania, która definiuje zasięg uchwytu, zostanie zakończona. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Operacja MQCLOSE uchwytu *HSUB* nie ma wpływu na uchwyt *HOBJ*.

Ten uchwyt nie może zostać przekazany do wywołania MQGET lub MQCB. Należy użyć parametru *HOBJ*. Przekazanie tego uchwytu do dowolnego innego wywołania programu IBM MQ spowoduje wywołanie RC2019.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Pomyślne zakończenie

CCWARN

Ostrzeżenie (częściowe zakończenie)

CCFAIL

Wywołanie zakończone niepowodzeniem

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2019

(2019 X'07E3') Uchwyt obiektu nie jest poprawny

RC2046

(2046 X'07FE') Opcje nie są poprawne lub nie są spójne

RC2085

(2085 X'0825 ') Nie można znaleźć zidentyfikowanego obiektu

RC2161

(2161 X'0871 ') Menedżer kolejek wygaszany

RC2298

(2298 X'08FA') Funkcja nie jest obsługiwana.

RC2424

(2424 X'0978 ') deskryptor subskrypcji (MQSD) nie jest poprawny

RC2425

(2441 X' 979 ') Łańcuch tematu nie jest poprawny

RC2428

(2428 X'097C') Podana nazwa subskrypcji nie jest zgodna z istniejącymi subskrypcjami.

RC2429

(2429 X'097D') Nazwa subskrypcji istnieje i jest używana przez inną aplikację.

RC2431

(2431 X'097F') Pole danych SubUser nie jest poprawne

RC2432

(2432 X'0980 ') Subskrypcja istnieje

RC2434

(2434 X'0982 ') Nazwa subskrypcji jest zgodna z istniejącą subskrypcją

RC2440

(2440 X'0988 ') Pole SubName jest niepoprawne

RC2441

(2441 X'0989 ') Pole Objectstring nie jest poprawne

RC2435

(2435 X'0983 ') Nie można zmienić atrybutu za pomocą SDALT lub subskrypcji utworzono za pomocą narzędzia SDIMM.

RC2436

(2436 X'0984 ') Opcja SODUR jest niepoprawna

RC2459

(2459, X'99B') Błąd składni łańcucha wyboru.

RC2503

(2503 X'09C7') wywołania MQSUB są obecnie zablokowane dla subskrybowanych tematów.

RC2519

(2519, X'9D7') Łańcuch wyboru nie jest określony w opisie sposobu korzystania ze struktury MQCHARV.

RC2551

(2551, X'9F7') Określony łańcuch wyboru nie jest dostępny.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQSUB(HCONN : SUBDSC : HOBJ :  
C                                HSUB : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQSUB      PR      EXTPROC('MQSUB')  
D* Connection handle  
D HCONN      10I 0 VALUE  
D* Subscription descriptor  
D SUBDSC      400A  
D* Object handle for queue  
D HOBJ      10I 0  
D* Subscription object handle  
D HSUB      10I 0  
D* Completion code  
D CMPCOD      10I 0  
D* Reason code qualifying CompCode  
D REASON      10I 0
```

MQSUBRQ-żądanie subskrypcji

Wywołanie MQSUBRQ składa się z żądania w subskrypcji.

- “Składnia” na stronie [1393](#)
- “Użycie notatek” na stronie [1393](#)
- “Parametry” na stronie [1394](#)
- “Deklaracja RPG” na stronie [1395](#)

Składnia

MQSUBRQ (*HCONN*, *HSUB*, *ACTION*, *SUBROPT*, *CMPCOD*, *REASON*)

Użycie notatek

Następujące uwagi dotyczące użycia mają zastosowanie do wykorzystania SRAPUB:

1. Jeśli komenda zakończy się pomyślnie, zachowane publikacje zgodne z podaną subskrypcją zostały wysłane do subskrypcji i można je odebrać za pomocą komendy MQGET lub MQCB, korzystając z komendy HOBJ zwróconej w oryginalnym czasowniku MQSUB, który utworzył subskrypcję.
2. Jeśli temat subskrybowany przez oryginalny komendę MQSUB, który utworzył subskrypcję, zawiera znak wieloznaczny, może zostać wysłana więcej niż jedna zachowana publikacja. Liczba publikacji wysłanych w wyniku tego wywołania jest rejestrowana w polu *SRNMP* w strukturze SBROPT.
3. Jeśli komenda zakończy działanie z kodem przyczyny RC2437, wówczas nie ma obecnie zachowanych publikacji dla określonego tematu.
4. Jeśli komenda zakończy działanie z kodem przyczyny RC2525 lub RC2526, w chwili obecnej istnieją zachowane publikacje dla określonego tematu, ale wystąpił błąd, który oznaczał, że nie można ich dostarczyć.
5. Aplikacja musi mieć bieżącą subskrypcję tematu, zanim będzie mogła wykonać to wywołanie. Jeśli subskrypcja została dokonana w poprzedniej instancji aplikacji, a poprawny uchwyt do subskrypcji nie

jest dostępny, aplikacja musi najpierw wywołać MQSUB z opcją SORES, aby uzyskać uchwyt do użycia w tym wywołaniu.

6. Publikacje są wysyłane do miejsca docelowego, które jest zarejestrowane w celu użycia z bieżącą subskrypcją tej aplikacji. Jeśli publikacje powinny zostać wysłane w innym miejscu, należy najpierw zmodyfikować subskrypcję przy użyciu wywołania MQSUB z opcją SOALT.

Parametry

Wywołanie MQSUBRQ ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS dla aplikacji CICS oraz w systemie IBM i dla aplikacji działających w trybie zgodności można pominąć wywołanie MQCONN, a także następującą wartość dla *HCONN* :

HCDEFH

Domyślny uchwyt połączenia.

Ważne:  Jeśli nadal jest używany tryb zgodności, należy rozważyć zrekompilowanie aplikacji i używanie wywołań powiązanych statycznie. Więcej informacji zawiera sekcja [Tryb kompatybilności dla aplikacji RPG w systemie IBM i](#)

HSUB (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje subskrypcję, dla której ma zostać zamówiona aktualizacja. Wartość *HSUB* została zwrócona z poprzedniego wywołania MQSUB.

ACTION (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten parametr steruje konkretnymi działaniami, które są żądane w subskrypcji. Należy określić jeden (i tylko jeden) z następujących elementów:

SRAPUB

To działanie wymaga, aby publikacja aktualizacji została wysłana dla określonego tematu. Jest to zwykle używane, jeśli subskrybent określił opcję SOPUBR w wywołaniu MQSUB podczas jego subskrybowania. Jeśli menedżer kolejek ma zachowaną publikację dla tematu, jest ona wysyłana do subskrybenta. Jeśli nie, wywołanie nie powiedzie się. Jeśli aplikacja jest wysyłana do publikacji, która została zachowana, jest ona wskazana przez właściwość komunikatu MQIsRetained w tej publikacji.

Ponieważ temat w istniejącej subskrypcji reprezentowanej przez parametr *HSUB* może zawierać znaki wieloznaczne, subskrybent może otrzymać wiele zachowanych publikacji.

SBROPT (MQSRO)-wejście/wyjście

Te opcje sterują działaniem MQSUBRQ, patrz [“MQSRO-opcje żądania subskrypcji” na stronie 592](#) , aby uzyskać szczegółowe informacje.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

CCOK

Pomyślne zakończenie

CCWARN

Ostrzeżenie (częściowe zakończenie)

CCFAIL

Wywołanie zakończone niepowodzeniem

Przyczyna (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CPMCO*D to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CPMCO*D to CCFAIL:

RC2298

2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

RC2437

2437 (X'0985 ') Nie ma zachowanych publikacji obecnie przechowywanych dla tego tematu.

RC2046

2046 (X'07FE') Parametr lub pole opcji zawiera opcje, które są niepoprawne, lub kombinacja opcji, która jest niepoprawna.

RC2161

2161 (X'0871 ') Wyciszanie menedżera kolejek

RC2438

2438 (X'0986 ') W wywołaniu MQSUBRQ opcje żądania subskrypcji MQSRO nie są poprawne.

Deklaracja RPG

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQSUBRQ(HCONN : HSUB : ACTION :  
C                                SBROPT : CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQSUBRQ      PR      EXTPROC('MQSUBRQ')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Subscription handle  
D HSUB              10I 0 VALUE  
D* Action requested on the subscription  
D ACTION            10I 0 VALUE  
D* Subscription Request Options  
D SBROPT            16A  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CompCode  
D REASON            10I 0
```

Atrybuty obiektów

Ta kolekcja tematów zawiera tylko te obiekty produktu IBM MQ, które mogą być przedmiotem wywołania funkcji MQINQ, a także podaje szczegółowe informacje na temat atrybutów, do których można się dowiedzieć, oraz do wybranych selektorów.

Atrybuty dla kolejek

Te informacje umożliwiają zapoznanie się z różnymi typami definicji kolejek i atrybutami obsługiwanymi przez poszczególne typy.

Typy kolejek: Menedżer kolejek obsługuje następujące typy definicji kolejek:

Kolejka lokalna

Jest to kolejka fizyczna, w której zapisywane są komunikaty. Kolejka istnieje w menedżerze kolejek lokalnych.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty i usuwać komunikaty z kolejek tego typu. Wartością atrybutu kolejki *QType* jest QTLOC.

Kolejka współużytkowana

Jest to kolejka fizyczna, w której zapisywane są komunikaty. Kolejka istnieje we współużytkowanym repozytorium, które jest dostępne dla wszystkich menedżerów kolejek należących do grupy współużytkowania kolejek, która jest właścicielem współużytkowanego repozytorium.

Aplikacje połączone z dowolnym menedżerem kolejek w grupie współużytkowania kolejki mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Takie kolejki są efektywnie takie same, jak kolejki lokalne. Wartością atrybutu kolejki *QType* jest QTLOC.

- Kolejki współużytkowane są obsługiwane tylko w systemie z/OS.

Kolejka klastra

Jest to kolejka fizyczna, w której zapisywane są komunikaty. Kolejka istnieje w lokalnym menedżerze kolejek lub na co najmniej jednym menedżerze kolejek, który należy do tego samego klastra, co lokalny menedżer kolejek.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu, niezależnie od miejsca położenia kolejki. Jeśli instancja kolejki istnieje w lokalnym menedżerze kolejek, kolejka zachowuje się w taki sam sposób, jak kolejka lokalna, a aplikacje połączone z lokalnym menedżerem kolejek mogą usuwać komunikaty z kolejki. Wartością atrybutu kolejki *QType* jest QTCLUS.

Kolejka aliasowa

Nie jest to kolejka fizyczna-jest to nazwa alternatywna dla kolejki lokalnej. Nazwa kolejki lokalnej, do której alias jest tłumaczona, jest częścią definicji kolejki aliasowej.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty i usuwać komunikaty z kolejek aliasowych-komunikaty są umieszczane i usuwane z kolejki lokalnej, do której alias jest tłumaczony. Wartością atrybutu kolejki *QType* jest QTALS.

Kolejka zdalna

Ta kolejka nie jest kolejką fizyczną-jest to lokalna definicja kolejki, która istnieje w zdalnym menedżerze kolejek. Lokalna definicja kolejki zdalnej zawiera informacje, które informują menedżera kolejek lokalnych, w jaki sposób kierować komunikaty do zdalnego menedżera kolejek.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach zdalnych-komunikaty są umieszczane w lokalnej kolejce transmisji używanej do kierowania komunikatów do zdalnego menedżera kolejek. Aplikacje nie mogą usuwać komunikatów z kolejek zdalnych. Wartością atrybutu kolejki *QType* jest QTREM.

Definicja kolejki zdalnej może być również używana dla:

- Aliasing kolejki odpowiedzi

W tym przypadku nazwą definicji jest nazwa kolejki odpowiedzi. Więcej informacji na ten temat zawiera sekcja [Definicje aliasów kolejki odpowiedzi](#).

- Aliasing menedżera kolejek

W tym przypadku nazwa definicji jest aliasem dla menedżera kolejek, a nie nazwą kolejki. Więcej informacji na ten temat zawiera sekcja [Definicje aliasów menedżera kolejek](#).

Kolejka modelowa

Nie jest to kolejka fizyczna-jest to zestaw atrybutów kolejki, z których można utworzyć kolejkę lokalną.

Komunikaty nie mogą być przechowywane w kolejkach tego typu.

Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek. Inne atrybuty kolejki mają zastosowanie tylko do określonych typów kolejek. Typy kolejek, do których ma zastosowanie atrybut, są oznaczane symbolem ✓ w Tabeli 176 na stronie 1397 i kolejnych tabelach.

Tabela 176 na stronie 1397 podsumowuje atrybuty, które są specyficzne dla kolejek. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy atrybutów wyświetlane w tej sekcji są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli komendy MQSC są używane do definiowania, modyfikowania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Szczegółowe informacje zawiera sekcja Komendy skryptowe (MQSC).

Tabela 176. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

Atrybut	Opis	Lokaln a	Model	Alias	Zdalny	Klaster
<u>AlterationDate</u>	Data ostatniej zmiany definicji	✓		✓	✓	
<u>AlterationTime</u>	Czas ostatniej zmiany definicji	✓		✓	✓	
<u>Nazwa QNameBackoutRequeue</u>	Nadmierna nazwa kolejki wycofanych komunikatów	✓	✓			
<u>BackoutThreshold</u>	Próg wycofania	✓	✓			
<u>BaseQName</u>	Nazwa kolejki, do której alias jest tłumaczący			✓		
<u>ClusterChannelNazwa</u>	Nazwa kanału nadawczego klastra	✓	✓			
<u>ClusterName</u>	Nazwa klastra, do którego należy kolejka	✓		✓	✓	
<u>ClusterNamelist</u>	Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka	✓		✓	✓	
<u>CreationDate</u>	Data utworzenia kolejki	✓				
<u>CreationTime</u>	Czas utworzenia kolejki	✓				
<u>CurrentQDepth</u>	Bieżąca głębokość kolejki	✓				
<u>DefBind</u>	Domyślne łączenie	✓		✓	✓	✓
<u>DefinitionType</u>	Typ definicji kolejki.	✓	✓			
<u>DefInputOpenOption</u>	Domyślna opcja otwarcia wejścia	✓	✓			
<u>DefPersistence</u>	Domyślna trwałość komunikatu	✓	✓	✓	✓	✓

Tabela 176. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

(kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
DefPriority	Domyślny priorytet komunikatu	✓	✓	✓	✓	✓
DistLists	Obsługa listy dystrybucyjnej	✓	✓			
HardenGetBackout	Czy zachować dokładną liczbę wycofań	✓	✓			
InhibitGet	Określa, czy operacje pobierania dla kolejki są dozwolone	✓	✓	✓		
InhibitPut	Określa, czy dozwolone są operacje put dla kolejki	✓	✓	✓	✓	✓
InitiationQName	Nazwa kolejki inicjuj.	✓	✓			
MaxMsgDługość	Maksymalna długość komunikatu w bajtach	✓	✓			
MaxQDepth	Maksymalna głębokość kolejki	✓	✓			
MediaLog	Tożsamość najstarszego przydziału dziennika (lub najstarszego dziennika w systemie IBM i) niezbędne do odtwarzania nośnika w określonej kolejce	✓	✓			
MsgDeliverySekuencja	Kolejność dostarczania komunikatów	✓	✓			
OpenInputLiczba	Liczba operacji otwierania dla danych wejściowych	✓				
OpenOutputLiczba	Liczba operacji otwierania dla danych wyjściowych	✓				

Tabela 176. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

(kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
ProcessName	Nazwa procesu	✓	✓			
ZdarzenieQDepthHigh	Określa, czy generowane są zdarzenia zapętnienia kolejki.	✓	✓			
QDepthHighLimit	Górny limit głębokości kolejki	✓	✓			
ZdarzenieQDepthLow	Określa, czy generowane są zdarzenia zapętnienia kolejki.	✓	✓			
QDepthLowLimit	Niski limit głębokości kolejki	✓	✓			
ZdarzenieQDepthMax	Określa, czy generowane są zdarzenia zapętnienia kolejki	✓	✓			
QDesc	Opis kolejki	✓	✓	✓	✓	✓
Nazwa QName	Nazwa kolejki	✓		✓	✓	✓
QServiceInterval	Cel dla przedziału czasu usługi kolejki	✓	✓			
QServiceIntervalZdarzenie	Określa, czy generowane są zdarzenia OK Odstęp czasu usługi lub Przedział czasu usługi OK	✓	✓			
QTYPE	Typ kolejki	✓		✓	✓	✓
RemoteQMgrNazwa	Nazwa zdalnego menedżera kolejek				✓	
RemoteQName	Nazwa kolejki zdalnej				✓	
RetentionInterval	Interwał przechowywania	✓	✓			

Tabela 176. Atrybuty dla kolejek. Kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych odnosi się również do kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można określić podczas otwierania kolejki klastra w celu uzyskania informacji o kolejce lub w celu uzyskania informacji i danych wyjściowych. Jeśli kolejka klastra jest otwarta dla zapytania plus jeden lub więcej danych wejściowych, przeglądania lub ustawiania, zamiast niej ma zastosowanie kolumna dla kolejek lokalnych.

(kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalny	Klaster
SCOPE	Określa, czy pozycja dla kolejki istnieje również w katalogu komórki.	✓		✓	✓	
Możliwość współużytkowania	Współużytkowalność kolejki	✓	✓			
TriggerControl	Kontrola wyzwalacza	✓	✓			
TriggerData	Dane wyzwalacza	✓	✓			
TriggerDepth	Wyzwalacz uruchamiany zapełnieniem	✓	✓			
PriorytetTriggerMsg	Próg priorytetu komunikatu dla wyzwalacza.	✓	✓			
TriggerType	Typ wyzwalacza	✓	✓			
Użycie	Wykorzystanie kolejki	✓	✓			
XmitQName	Nazwa kolejki transmisji				✓	

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami na końcu, aby długość 12 bajtów (na przykład 1992-09-23↵↵, gdzie ↵↵ oznacza dwa puste znaki).

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) są zmieniane w miarę działania menedżera kolejek. Zmiany w tych atrybutach nie mają wpływu na *AlterationDate*.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Jest to czas ostatniej zmiany definicji. Format czasu to HH.MM.SS , przy użyciu zegara 24-godzinnego, z zerowym zerem, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20). Jest to czas lokalny.

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) są zmieniane w miarę działania menedżera kolejek. Zmiany w tych atrybutach nie mają wpływu na *AlterationTime*.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podawana przez LNTIME.

BackoutRequeueQName (48-bajtowy łańcuch znaków)

Nadmierna nazwa kolejki wycofanych komunikatów.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Aplikacje działające w produkcie WebSphere Application Server i te, które korzystają z narzędzi serwera aplikacji produktu IBM MQ , używają tego atrybutu do określenia, gdzie powinny być wyświetlane komunikaty, których kopie zapasowe zostały wycofane. W przypadku wszystkich innych aplikacji, oprócz zezwolenia na odpytywanie, menedżer kolejek nie podejmuje żadnych działań na podstawie wartości atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CABRQN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

BackoutThreshold (10-cyfrowa liczba całkowita ze znakiem)

Próg wycofania.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Aplikacje działające w produkcie WebSphere Application Server i te, które korzystają z narzędzi serwera aplikacji IBM MQ , używają tego atrybutu do określenia, czy należy utworzyć kopię zapasową komunikatu. W przypadku wszystkich innych aplikacji, oprócz zezwolenia na odpytywanie, menedżer kolejek nie podejmuje żadnych działań na podstawie wartości atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora IABTHR z wywołaniem MQINQ.

BaseQName (48-bajtowy łańcuch znaków)

Nazwa kolejki, do której alias jest tłumaczona.

Lokalna	Model	Alias	Zdalny	Klaster
		✓		

Jest to nazwa kolejki, która jest zdefiniowana dla lokalnego menedżera kolejek. (Więcej informacji na temat nazw kolejek znajduje się w opisie pola *ODON* w tabeli MQOD. Kolejka jest jednym z następujących typów:

QTLOC

Kolejka lokalna.

QTREM

Lokalna definicja kolejki zdalnej.

QTCLUS

Kolejka klastra.

Aby określić wartość tego atrybutu, należy użyć selektora CABASQ z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

BaseType (struktura parametru liczby całkowitej)

Typ obiektu, do którego alias jest rozstrzygany.

Lokalna	Model	Alias	Zdalny	Klaster
		✓		

Ten atrybut może mieć jedną z następujących wartości:

OTQ

Podstawowy typ obiektu to kolejka

OTTOP

Podstawowy typ obiektu to temat

CFStrucName (12-bajtowy łańcuch znaków)

Nazwa struktury narzędzia CF.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to nazwa struktury narzędzia CF, w której zapisywane są komunikaty w kolejce. Pierwszy znak nazwy mieści się w zakresie od A do Z, a pozostałe znaki są w zakresie od A do Z, od 0 do 9 lub puste.

Pełna nazwa struktury w narzędziu CF jest uzyskiwane poprzez przyrostek wartości atrybutu menedżera kolejek produktu *QSGName* z wartością atrybutu kolejki produktu *CFStrucName*.

Ten atrybut ma zastosowanie tylko do współużytkowanych kolejek; jest ignorowany, jeśli *QSGDisp* nie ma wartości *QSGDSH*.

Aby określić wartość tego atrybutu, należy użyć selektora CACFSN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNCFSN.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

ClusterChannelNazwa (dwudziestobajtowy łańcuch znaków)

ClusterChannelNazwa to ogólna nazwa kanałów nadawczych klastra, które używają tej kolejki jako kolejki transmisji. Atrybut określa, które kanały nadawcze klastra wysyłają komunikaty do kanału odbiorczego klastra z tej kolejki transmisji klastra.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Menedżer kolejek jest domyślnie skonfigurowany w taki sposób, aby wszystkie kanały nadawcze klastra wysyłały komunikaty z pojedynczej kolejki transmisji: *SYSTEM.CLUSTER.TRANSMIT.QUEUE*.

Konfigurację domyślną można zmienić, modyfikując atrybut **DefClusterXmitQueueType** menedżera kolejek. Wartością domyślną tego atrybutu jest SCTQ. Wartość tę można zmienić na CHANNEL. Jeśli atrybut **DefClusterXmitQueueType** zostanie ustawiony na wartość CHANNEL, dla każdego kanału nadawczego klastra domyślnie zostanie użyta konkretna kolejka transmisji klastra `SYSTEM.CLUSTER.TRANSMIT.ChannelName`.

Kanał nadawczy klastra dla atrybutu `ClusterChannelName` kolejki transmisji można również ustawić ręcznie. Komunikaty przeznaczone dla menedżera kolejek połączonego kanałem nadawczym klastra są przechowywane w kolejce transmisji identyfikującej kanał nadawczy klastra. Nie są one przechowywane w domyślnej kolejce transmisji klastra. Jeśli dla atrybutu `ClusterChannelName` zostaną ustawione wartości puste, po zrestartowaniu kanału zostanie on przetaczony na domyślną kolejkę transmisji klastra. Kolejka domyślna to `SYSTEM.CLUSTER.TRANSMIT.ChannelName` lub `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, w zależności od wartości atrybutu `DefClusterXmitQueueType` menedżera kolejek.

Określenie w atrybucie **ClusterChannelName** gwiazdek ("*") umożliwia powiązanie kolejki transmisji z zestawem kanałów nadawczych klastra. Gwiazdki mogą znajdować się na początku, na końcu lub na dowolnej liczbie miejsc w środku łańcucha nazwy kanału. Długość atrybutu **ClusterChannelName** jest ograniczona do 20 znaków: `MQ_CHANNEL_NAME_LENGTH`.

ClusterName (48-bajtowy łańcuch znaków)

Nazwa klastra, do którego należy kolejka.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Jest to nazwa klastra, do którego należy kolejka. Jeśli kolejka należy do więcej niż jednego klastra, `ClusterNameList` określa nazwę obiektu listy nazw, która identyfikuje klastry, a `ClusterName` jest pusta. Co najmniej jedna z wartości `ClusterName` i `ClusterNameList` musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CACLN przy użyciu wywołania MQINQ. Długość tego atrybutu jest podana przez LNCLUN.

ClusterNameList (48-bajtowy łańcuch znaków)

Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Jest to nazwa obiektu listy nazw, który zawiera nazwy klastrów, do których należy ta kolejka. Jeśli kolejka należy do tylko jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć opcji `ClusterName` do określenia nazwy klastra, w którym to przypadku pole `ClusterNameList` jest puste. Co najmniej jedna z wartości `ClusterName` i `ClusterNameList` musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CACLNL z wywołaniem MQINQ. Długość tego atrybutu jest nadawana przez LNNLN.

CreationDate (12-bajtowy łańcuch znaków)

Data utworzenia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓				

Data utworzenia kolejki. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami na końcu, aby długość 12 bajtów (na przykład 1992-09-23-- , gdzie -- oznacza dwa puste znaki).

- W systemie IBM data utworzenia kolejki może różnić się od daty bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CACRTD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNCRTD.

CreationTime (8-bajtowy łańcuch znaków)

Czas utworzenia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓				

Jest to czas utworzenia kolejki. Format czasu to HH.MM.SS , przy użyciu zegara 24-godzinnego, z zerowym zerem, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20). Jest to czas lokalny.

- W systemie IBM czas utworzenia kolejki może różnić się od czasu utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CACRTT z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNCRTT.

CurrentQDepth (10-cyfrowa liczba całkowita ze znakiem)

Bieżące zapętnienie kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓				

To jest liczba komunikatów znajdujących się aktualnie w kolejce. Wartość ta jest zwiększana podczas wywołania MQPUT i podczas wycofywania wywołania MQGET. Jest ono zmniejszane podczas wywołania MQGET bez przeglądania i podczas wycofywania wywołania MQPUT. Wynika to z faktu, że liczba obejmuje komunikaty, które zostały umieszczone w kolejce w ramach jednostki pracy, ale które nie zostały jeszcze zatwierdzone, nawet jeśli nie są zakwalifikowane do pobrania za pomocą wywołania MQGET. Podobnie, nie obejmuje ona komunikatów, które zostały pobrane w ramach jednostki pracy przy użyciu wywołania MQGET, ale które nie zostały jeszcze zatwierdzone.

Liczba ta obejmuje również komunikaty, które przeszły czas utraty ważności, ale nie zostały jeszcze odrzucone, mimo że te komunikaty nie są zakwalifikowane do pobrania. Zapoznaj się z polem *MDEXP* , które opisano w sekcji [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Przetwarzanie jednostkowe i segmentacja komunikatów może spowodować, że *CurrentQDepth* przekroczy *MaxQDepth*. Nie wpływa to jednak na możliwość pobierania komunikatów- wszystkie komunikaty w kolejce mogą być pobierane za pomocą wywołania MQGET w normalny sposób.

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IACDEP przy użyciu wywołania MQINQ.

DefBind (10-cyfrowa liczba całkowita ze znakiem)

Powiązanie domyślne.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	✓

Ten atrybut jest domyślnym powiązaniem, który jest używany, gdy parametr OOBNDQ jest określony w wywołaniu MQOPEN, a kolejka jest kolejką klastra. DefBind może mieć jedną z następujących wartości:

BNDOPN

Powiązanie ustalone przez wywołanie MQOPEN.

BNDNOT

Powiązanie nie zostało ustalone.

BNDGRP

Powiązanie nie jest naprawiane przez wywołanie MQOPEN, ale jest stałe dla MQPUT dla wszystkich komunikatów w grupie logicznej.

Aby określić wartość tego atrybutu, należy użyć selektora IADBND przy użyciu wywołania MQINQ.

DefinitionType (10-cyfrowa liczba całkowita ze znakiem)

Typ definicji kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Wskazuje, w jaki sposób została zdefiniowana kolejka. Wartość ta jest jedną z następujących wartości:

QDPRE

Predefiniowana kolejka stała.

Kolejka jest stałą kolejką utworzoną przez administratora systemu. Tylko administrator systemu może go usunąć.

Predefiniowane kolejki są tworzone za pomocą komendy MQSC DEFINE i mogą być usuwane tylko za pomocą komendy MQSC DELETE . Predefiniowanych kolejek nie można tworzyć z kolejek modelowych.

Komendy mogą być wydawane przez operatora lub przez uprawnionego użytkownika wysyłającego komunikat komendy do kolejki wejściowej komend (patrz atrybut *CommandInputQName* opisany w sekcji [“Atrybuty dla menedżera kolejek”](#) na stronie 1427).

QDPERM

Dynamicznie zdefiniowana kolejka stała.

Kolejka to kolejka trwała, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. W definicji kolejki modelowej wartość QDPERM jest określona dla atrybutu *DefinitionType* .

Ten typ kolejki można usunąć przy użyciu wywołania MQCLOSE. Więcej szczegółów na ten temat zawiera sekcja [“MQCLOSE-zamknięcie obiektu”](#) na stronie 1288.

Wartością atrybutu *QSGDIsP* dla trwałej kolejki dynamicznej jest QSGDQM.

QDTEMP

Dynamicznie zdefiniowana kolejka tymczasowa.

Kolejka jest kolejką tymczasową, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. W definicji kolejki modelowej wartość QDTEMP jest określona dla atrybutu *DefinitionType* .

Ten typ kolejki jest automatycznie usuwany przez wywołanie MQCLOSE, gdy jest on zamykany przez aplikację, która go utworzyła.

Wartością atrybutu *QSGDisp* dla tymczasowej kolejki dynamicznej jest QSGDQM.

QDSHAR

Dynamicznie zdefiniowana kolejka współużytkowana.

Kolejka jest współużytkowaną statą kolejką, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskrytorze obiektu MQOD. Definicja kolejki modelowej miała wartość QDSHAR dla atrybutu *DefinitionType*.

Ten typ kolejki można usunąć przy użyciu wywołania MQCLOSE. Więcej szczegółów na ten temat zawiera sekcja [“MQCLOSE-zamknięcie obiektu”](#) na stronie 1288.

Wartością atrybutu *QSGDisp* dla współużytkowanej kolejki dynamicznej jest QSGDSH.

Ten atrybut w definicji kolejki modelowej nie wskazuje, w jaki sposób została zdefiniowana kolejka modelowa, ponieważ kolejki modelowe są zawsze predefiniowane. Zamiast tego wartość tego atrybutu w kolejce modelowej jest używana do określenia *DefinitionType* każdej kolejki dynamicznej utworzonej z definicji kolejki modelowej przy użyciu wywołania MQOPEN.

Aby określić wartość tego atrybutu, należy użyć selektora IADEF z wywołaniem MQINQ.

DefInputOpenOption (10-cyfrowa liczba całkowita ze znakiem)

Domyślna opcja otwierania danych wejściowych.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to domyślny sposób, w jaki kolejka powinna być otwarta dla danych wejściowych. Ma zastosowanie, jeśli podczas otwierania kolejki opcja OOINPQ jest określona w wywołaniu MQOPEN. Może to mieć jedną z następujących wartości:

OOINPX

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie nie powiodło się z kodem przyczyny RC2042, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla danych wejściowych dowolnego typu (OOINPS lub OOINPX).

OOINPS

Otwórz kolejkę, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

Kolejka jest otwierana do użycia z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest aktualnie otwarta przez tę lub inną aplikację z OOINPS, ale kończy się niepowodzeniem z kodem przyczyny RC2042, jeśli kolejka jest obecnie otwarta z OOINPX.

Aby określić wartość tego atrybutu, należy użyć selektora IADINP z wywołaniem MQINQ.

DefPersistence (10-cyfrowa liczba całkowita ze znakiem)

Domyślna trwałość komunikatu.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓	✓	✓

Jest to domyślna trwałość komunikatów w kolejce. Ma ona zastosowanie, jeśli w deskrytorze komunikatu określono parametr PEQDEF, gdy komunikat jest umieszczany.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w definicji *pierwszej* w ścieżce w czasie wywołania MQPUT lub MQPUT1. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Może to mieć jedną z następujących wartości:

PEPER

Komunikat jest trwały.

Oznacza to, że komunikat jest zachowany po awariach systemu i restartach menedżera kolejek. Komunikaty trwałe nie mogą być umieszczane w następujących systemach:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane

Komunikaty trwałe mogą być umieszczane w statycznych kolejkach dynamicznych i predefiniowanych kolejkach.

PENPER

Komunikat nie jest trwały.

Oznacza to, że zwykle komunikat nie jest w stanie przetrwać awarii systemu lub restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartu menedżera kolejek zostanie znaleziona nienaruszona kopia komunikatu w pamięci dyskowej.

W specjalnym przypadku kolejek współużytkowanych komunikaty nietrwałe *do* przeżywają restarty menedżerów kolejek w grupie współużytkowania kolejki, ale nie przeżywają niepowodzeń narzędzia CF używanego do przechowywania komunikatów w kolejkach współużytkowanych.

Zarówno komunikaty trwałe, jak i nietrwałe mogą istnieć w tej samej kolejce.

Aby określić wartość tego atrybutu, należy użyć selektora IADPER z wywołaniem MQINQ.

DefPriority (10-cyfrowa liczba całkowita ze znakiem)

Domyślny priorytet komunikatu.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓	✓	✓

Jest to domyślny priorytet komunikatów w kolejce. Ma to zastosowanie, jeśli w deskrypcorze komunikatu określono wartość PRQDEF, gdy komunikat jest umieszczany w kolejce.

Jeśli w ścieżce rozstrzygania nazw kolejek istnieje więcej niż jedna definicja, domyślny priorytet komunikatu jest przyjmowany z wartości tego atrybutu w definicji *pierwszej* podanej w ścieżce w czasie operacji put. Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład kolejka *DefXmitQName*)

Sposób, w jaki komunikat jest umieszczany w kolejce, zależy od wartości atrybutu *MsgDeliverySequence* kolejki:

- Jeśli atrybut *MsgDeliverySequence* jest atrybutem MSPRIO, to pozycja logiczna, w której komunikat jest umieszczany w kolejce, zależy od wartości pola *MDPRI* w deskrypcorze komunikatu.

- Jeśli atrybut *MsgDeliverySequence* jest atrybutem MSFIFO, komunikaty są umieszczane w kolejce tak, jakby miały priorytet równy *DefPriority* rozstrzygniętej kolejki, niezależnie od wartości pola *MDPRI* w deskrytorze komunikatu. Jednak pole *MDPRI* zachowuje wartość określoną przez aplikację, która wstawiła komunikat. Więcej informacji na ten temat zawiera opis atrybutu *MsgDeliverySequence* opisany w sekcji “Atrybuty dla kolejek” na stronie 1395.

Priorytety są w zakresie od zera (od najniższego) do *MaxPriority* (najwyższy); patrz atrybut *MaxPriority* opisany w sekcji “Atrybuty dla menedżera kolejek” na stronie 1427.

Aby określić wartość tego atrybutu, należy użyć selektora IADPRI przy użyciu wywołania MQINQ.

DefReadAhead (10-cyfrowa liczba całkowita ze znakiem)

Określa domyślne zachowanie odczytu z wyprzedzeniem dla nietrwałych komunikatów dostarczanych do klienta.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓		

DefReadAhead można ustawić na jedną z następujących wartości:

RAHNO

Komunikaty nietrwałe nie są wysyłane z wyprzedzeniem do klienta przed ich żądaniem. Jeśli działanie klienta zostanie zakończone nieprawidłowo, może zostać utracony maksymalnie jeden komunikat nietrwały.

RAHTAK

Komunikaty nietrwałe są wysyłane z wyprzedzeniem do klienta, zanim aplikacja je zażąda. Komunikaty nietrwałe mogą zostać utracone, jeśli klient zakończy się nieprawidłowo lub jeśli klient nie zużywa wszystkich wysłanych wiadomości.

RAHDIS

Odczyt z wyprzedzeniem dla nietrwałych komunikatów, które nie zostały włączone dla tej kolejki. Komunikaty nie są wysyłane z wyprzedzeniem do klienta niezależnie od tego, czy aplikacja kliencka żąda odczytu z wyprzedzeniem.

Aby określić wartość tego atrybutu, należy użyć selektora IADRAH przy użyciu wywołania MQINQ.

DefPResp (10-cyfrowa liczba całkowita ze znakiem)

Domyślny atrybut typu put odpowiedzi (put response type-DEFPRESP) definiuje wartość używaną przez aplikację, gdy typ PutResponsew MQPMO został ustawiony na PMRASQ. Ten atrybut jest poprawny dla wszystkich typów kolejek.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓	✓	✓

Może to mieć jedną z następujących wartości:

SYNCHRONICZNY

Operacja put jest wydawana synchronicznie, zwracając odpowiedź.

ASYNCHRONICZNY

Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

Aby określić wartość tego atrybutu, należy użyć selektora IADPRT z wywołaniem MQINQ.

DistLists (10-cyfrowa liczba całkowita ze znakiem)

Obsługa listy dystrybucyjnej.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Wskazuje, czy komunikaty listy dystrybucyjnej mogą być umieszczane w kolejce. Atrybut jest ustawiany przez agenta kanału komunikatów (MCA) w celu poinformowania lokalnego menedżera kolejek, czy menedżer kolejek na drugim końcu kanału obsługuje listy dystrybucyjne. Ten ostatni menedżer kolejek (zwany "menedżerem kolejek partneringowych") jest tym, który następnie otrzymuje komunikat po usunięciu go z lokalnej kolejki transmisji przez wysyłający agent MCA.

Atrybut jest ustawiany przez wysyłający agent MCA za każdym razem, gdy nawiązuje połączenie z odbierającym MCA w partnerskim menedżerze kolejek. W ten sposób wysyłający agent MCA może spowodować, że lokalny menedżer kolejek będzie umieszczać w kolejce transmisji komunikaty, które menedżer kolejek może przetwarzać poprawnie.

Ten atrybut jest przeznaczony przede wszystkim do użycia z kolejkami transmisji, ale opisywane przetwarzanie jest wykonywane niezależnie od użycia zdefiniowanego dla kolejki (patrz atrybut *Usage*).

Może to mieć jedną z następujących wartości:

DLSUPP

Obsługiwane są listy dystrybucyjne.

Oznacza to, że komunikaty listy dystrybucyjnej mogą być zapisywane w kolejce i przesyłane do menedżera kolejek partnerskiego w tej postaci. Zmniejsza to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych.

DLNSUP

Listy dystrybucyjne nie są obsługiwane.

Oznacza to, że komunikaty z listy dystrybucyjnej nie mogą być zapisywane w kolejce, ponieważ menedżer kolejek partnerujących nie obsługuje list dystrybucyjnych. Jeśli aplikacja umieszcza komunikat z listą dystrybucyjną i ten komunikat ma zostać umieszczony w tej kolejce, menedżer kolejek rozdziela komunikat listy dystrybucyjnej i umieszcza poszczególne komunikaty w kolejce zamiast tego komunikatu. Zwiększa to ilość przetwarzania wymaganą do wysłania komunikatu do wielu miejsc docelowych, ale zapewnia, że komunikaty będą przetwarzane poprawnie przez partnerski menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IADIST za pomocą wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

HardenGetBackout (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy ma być zachowana dokładna liczba wycofań.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Dla każdego komunikatu jest zachowana liczba określająca, ile razy komunikat jest pobierany przez wywołanie MQGET w ramach jednostki pracy i że jednostka pracy została później wycofana. Liczba ta jest dostępna w polu *MDBOC* w deskrypcorze komunikatu po zakończeniu wywołania MQGET.

Licznik wycofań komunikatów jest zachowany przy restartowaniu menedżera kolejek. Aby jednak upewnić się, że liczba jest dokładna, informacje muszą być "utwardzone" (rejestrowane na dysku lub innym stałym urządzeniu pamięci masowej) za każdym razem, gdy komunikat jest pobierany za pomocą wywołania MQGET w ramach jednostki pracy dla tej kolejki. Jeśli ta opcja nie zostanie wykonana, a awaria menedżera kolejek zostanie wykonana razem z wycofaniem wywołania MQGET, licznik może nie być zwiększany.

Utwardzanie informacji dla każdego wywołania MQGET w jednostce pracy narzuca jednak koszt wydajności, a atrybut *HardenGetBackout* powinien być ustawiony na wartość QABH tylko wtedy, gdy liczba ta musi być dokładna.

- W systemie IBM iliczba wycofanych komunikatów jest zawsze utwardzana, niezależnie od ustawienia tego atrybutu.

Dozwolone są następujące wartości:

QABH

Zapamiętana liczba wycofań.

Hartowanie jest używane w celu zapewnienia, że liczba wycofań komunikatów w tej kolejce jest dokładna.

QABNH

Liczba wycofań może nie zostać zapamiętana.

Utwardzanie nie jest używane, aby upewnić się, że liczba wycofań komunikatów w tej kolejce jest dokładna. W związku z tym liczba ta może być mniejsza niż powinna.

Aby określić wartość tego atrybutu, należy użyć selektora IAHGB przy użyciu wywołania MQINQ.

InhibitGet (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy operacje pobierania dla tej kolejki są dozwolone.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓		

Jeśli kolejka jest kolejką aliasową, operacje get muszą być dozwolone zarówno dla aliasu, jak i dla kolejki podstawowej w czasie operacji pobierania, aby wywołanie MQGET powiodło się. Wartość ta jest jedną z następujących wartości:

QAGETI

Operacje pobierania są zablokowane.

Wywołania MQGET nie powiodły się z kodem przyczyny RC2016. Dotyczy to wywołań MQGET, które określają parametr GMBRWF lub GMBRWN.

Uwaga: Jeśli wywołanie MQGET działające w ramach jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu *InhibitGet* po wartości QAGETI nie uniemożliwia wykonania jednostki pracy.

QAGETA

Operacje pobierania są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora IAIGET z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InhibitPut (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy operacje put dla tej kolejki są dozwolone.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓	✓	✓

Jeśli w ścieżce rozstrzygania nazw kolejek istnieje więcej niż jedna definicja, należy zezwolić na operacje put dla *wszystkich* definicji w ścieżce (w tym definicji aliasów menedżera kolejek) w czasie operacji put, aby wywołanie MQPUT lub MQPUT1 powiodło się. Może to mieć jedną z następujących wartości:

QAPUTI

Operacje put są zablokowane.

Wywołania MQPUT i MQPUT1 kończą się niepowodzeniem z kodem przyczyny RC2051.

Uwaga: Jeśli wywołanie MQPUT działające w obrębie jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu *InhibitPut* w późniejszym czasie na QAPUTI nie uniemożliwia jednostkowej pracy zatwierdzonej pracy.

QAPUTA

Operacje put są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora IAIPUT przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InitiationQName (48-bajtowy łańcuch znaków)

Nazwa kolejki inicjuj.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to nazwa kolejki zdefiniowanej w lokalnym menedżerze kolejek. Kolejka musi być typu QTLOC. Menedżer kolejek wysyła komunikat wyzwalacza do kolejki inicjuj, gdy uruchomienie aplikacji jest wymagane w wyniku komunikatu przybywającego do kolejki, do której należy ten atrybut. Kolejka inicjująca musi być monitorowana przez aplikację monitora wyzwalacza, która uruchomi odpowiednią aplikację po odebraniu komunikatu wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora CAINIQ z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

MaxMsgDługość (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna długość komunikatu w bajtach.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to górna granica długości najdłuższego komunikatu *fizycznego*, który może zostać umieszczony w kolejce. Jednak ponieważ atrybut kolejki *MaxMsgLength* może być ustawiony niezależnie od atrybutu menedżera kolejek produktu *MaxMsgLength*, rzeczywisty górny limit długości najdłuższego komunikatu fizycznego, który może być umieszczony w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, istnieje możliwość umieszczenia komunikatu *logicznego*, który jest dłuższy niż mniejszy z dwóch atrybutów produktu *MaxMsgLength*, ale tylko wtedy, gdy aplikacja określa flagę MFSEGA w strukturze MQMD. Jeśli ta opcja jest określona, górna granica długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zwykle ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym aplikacja jest uruchomiona, powoduje zmniejszenie dolnego limitu.

Próba umieszczenia w kolejce komunikatu, który jest zbyt długi, kończy się niepowodzeniem z kodem przyczyny:

- RC2030, jeśli komunikat jest zbyt duży dla kolejki
- RC2031, jeśli komunikat jest zbyt duży dla menedżera kolejek, ale nie jest zbyt duży dla kolejki.

Dolny limit dla atrybutu *MaxMsgLength* wynosi zero. Górna granica jest określana przez środowisko:

- W systemie IBM maksymalna długość komunikatu wynosi 100 MB (104 857 600 bajtów).

Więcej informacji na ten temat zawiera opis parametru *BUFLen* opisanego w sekcji “MQPUT-umieszczanie komunikatu” na stronie 1358.

Aby określić wartość tego atrybutu, należy użyć selektora IAMLEN przy użyciu wywołania MQINQ.

MaxQDepth (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna głębokość kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to zdefiniowany górny limit dla liczby komunikatów fizycznych, które mogą istnieć w kolejce w dowolnym momencie. Próba umieszczenia komunikatu w kolejce, która zawiera już komunikaty produktu *MaxQDepth*, nie powiodła się. Kod przyczyny: RC2053.

Przetwarzanie jednostkowe i segmentacja komunikatów może spowodować, że rzeczywista liczba komunikatów fizycznych w kolejce przekracza *MaxQDepth*. Nie wpływa to jednak na możliwość pobierania komunikatów- wszystkie komunikaty w kolejce mogą być pobierane za pomocą wywołania MQGET w normalny sposób.

Wartość tego atrybutu jest równa zero lub większa. Górna granica jest określana przez środowisko.

Uwaga: Ilość miejsca w pamięci masowej dostępnego dla kolejki może być wyczerpana, nawet jeśli w kolejce znajduje się mniej niż *MaxQDepth* komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora IAMDEP z wywołaniem MQINQ.

MediaLog (10-cyfrowa liczba całkowita ze znakiem)

Tożsamość zakresu dziennika (lub dziennika w systemie IBM i) niezbędne do odtwarzania nośnika w określonej kolejce.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

W przypadku menedżerów kolejek, w których jest używane rejestrowanie cykliczne, wartość jest zwracana jako łańcuch o wartości NULL.

Sekwencja MsgDelivery(10-cyfrowa liczba całkowita ze znakiem)

Kolejność dostarczania komunikatów.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Określa kolejność, w jakiej komunikaty są zwracane do aplikacji przy użyciu wywołania MQGET:

MSFIFO

Komunikaty są zwracane w kolejności FIFO (najpierw w kolejności, w pierwszej kolejności).

Oznacza to, że wywołanie MQGET zwróci komunikat *first*, który spełnia kryteria wyboru określone w wywołaniu, niezależnie od priorytetu komunikatu.

MSPRIO

Komunikaty są zwracane w kolejności priorytetów.

Oznacza to, że wywołanie MQGET zwróci komunikat *highest-priority*, który spełnia kryteria wyboru określone w wywołaniu. W ramach każdego poziomu priorytetu komunikaty są zwracane w kolejności FIFO (najpierw w kolejności, w pierwszej kolejności).

Jeśli odpowiednie atrybuty zostaną zmienione w czasie, gdy w kolejce znajdują się komunikaty, kolejność dostarczania jest następująca:

- Kolejność, w jakiej komunikaty są zwracane przez wywołanie MQGET, jest określana na podstawie wartości atrybutów *MsgDeliverySequence* i *DefPriority*, które są wymuszane dla kolejki w momencie, gdy komunikat jest wyświetlany w kolejce:
 - Jeśli *MsgDeliverySequence* to MSFIFO po nadejściu komunikatu, komunikat jest umieszczany w kolejce tak, jakby jego priorytetem było *DefPriority*. Nie ma to wpływu na wartość pola *MDPRI* w deskrypcji komunikatu komunikatu. Pole to zachowuje wartość, jaką miała podczas pierwszego umieszczania komunikatu.
 - Jeśli *MsgDeliverySequence* jest MSPRIO po nadejściu komunikatu, komunikat jest umieszczany w kolejce w miejscu właściwym dla priorytetu podanego w polu *MDPRI* w deskrypcji komunikatu.

Jeśli wartość atrybutu *MsgDeliverySequence* zostanie zmieniona w czasie, gdy w kolejce znajdują się komunikaty, kolejność komunikatów w kolejce nie zostanie zmieniona.

Jeśli wartość atrybutu *DefPriority* zostanie zmieniona w czasie, gdy w kolejce znajdują się komunikaty, komunikaty nie muszą być dostarczane w kolejności FIFO, nawet jeśli atrybut *MsgDeliverySequence* jest ustawiony na wartość MSFIFO. Te komunikaty, które zostały umieszczone w kolejce z wyższym priorytetem, są dostarczane jako pierwsze.

Aby określić wartość tego atrybutu, należy użyć selektora IAMDS z wywołaniem MQINQ.

Liczba OpenInput(10-cyfrowa liczba całkowita ze znakiem)

Liczba operacji otwierania dla danych wejściowych.

Lokalna	Model	Alias	Zdalny	Klaster
✓				

Jest to liczba uchwytów, które są obecnie poprawne w przypadku usuwania komunikatów z kolejki przy użyciu wywołania MQGET. Jest to łączna liczba takich uchwytów znanych z *lokalnego* menedżera kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba nie obejmuje otwierania danych wejściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

Liczba ta obejmuje uchwyt, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla wejścia. Liczba ta nie obejmuje uchwytów, w których kolejka została otwarta dla działań, które nie zawierają danych wejściowych (na przykład kolejka otwarta tylko do przeglądania).

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IAOIC przy użyciu wywołania MQINQ.

Liczba OpenOutput(10-cyfrowa liczba całkowita ze znakiem)

Liczba operacji otwierania dla danych wyjściowych.

Lokalna	Model	Alias	Zdalny	Klaster
✓				

Jest to liczba uchwytów, które są obecnie poprawne w przypadku dodawania komunikatów do kolejki przy użyciu wywołania MQPUT. Jest to łączna liczba takich uchwytów znanych dla *lokalnego* menedżera kolejek. Nie obejmuje ona otwierania danych wyjściowych, które zostały wykonane dla tej kolejki

w zdalnych menedżerach kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba nie obejmuje operacji otwierania dla danych wyjściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejki, do której należy lokalny menedżer kolejek.

Liczba ta obejmuje uchwyt, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla danych wyjściowych. Liczba ta nie obejmuje uchwytów, w których kolejka została otwarta dla działań, które nie zawierają danych wyjściowych (na przykład kolejka otwarta tylko dla zapytania).

Wartość tego atrybutu zmienia się w zależności od tego, czy działa menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IA00C w wywołaniu MQINQ.

ProcessName (48-bajtowy łańcuch znaków)

Nazwa procesu.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to nazwa obiektu procesu, który jest zdefiniowany w menedżerze kolejek lokalnych. Obiekt procesu identyfikuje program, który może serwisować kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CAPRON z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNPRON.

QDepthHighZdarzenie (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia zapełnienia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Zdarzenie Wysokie zapełnienie kolejki wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wartości progowej zapełnienia kolejki (patrz atrybut *QDepthHighLimit*).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

QDepthHighZdarzenie może mieć jedną z dwóch wartości:

EVARDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDHE z wywołaniem MQINQ.

QDepthHighLimit (10-cyfrowa liczba całkowita ze znakiem)

Górny limit głębokości kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to wartość progowa, względem której porównywana jest głębokość kolejki w celu wygenerowania zdarzenia o głębokości kolejki. To zdarzenie wskazuje, że aplikacja umieściła komunikat w kolejce, co

spowodowało, że liczba komunikatów w kolejce stała się większa niż lub równa wartości progowej zapętnienia kolejki. Patrz atrybut *QDepthHighEvent*.

Wartość jest wyrażona jako wartość procentowa maksymalnej głębokości kolejki (atrybut *MaxQDepth*) i jest w zakresie od zera do 100. Wartość domyślna to 80.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDHL przy użyciu wywołania MQINQ.

QDepthLowZdarzenie (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia zapętnienia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Zdarzenie Niskie zapętnienie kolejki wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnego progu głębokości kolejki (patrz atrybut *QDepthLowLimit*).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

QDepthLowZdarzenie może mieć jedną z następujących wartości:

EVREDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IAQDLE z wywołaniem MQINQ.

QDepthLowLimit (10-cyfrowa liczba całkowita ze znakiem)

Niski limit głębokości kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to wartość progowa, względem której porównywana jest głębokość kolejki w celu wygenerowania zdarzenia niedobr kolejki. To zdarzenie wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnego progu głębokości kolejki. Patrz atrybut *QDepthLowEvent*.

Wartość jest wyrażona jako wartość procentowa maksymalnej głębokości kolejki (atrybut *MaxQDepth*) i jest w zakresie od zera do 100. Wartością domyślną jest 20.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDLL z wywołaniem MQINQ.

QDepthMaxZdarzenie (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia zapętnienia kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Zdarzenie zapętnienia kolejki wskazuje, że żądanie umieszczenia w kolejce zostało odrzucone, ponieważ kolejka jest pełna, to znaczy, że głębokość kolejki osiągnęła już maksymalną wartość.

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Może to mieć jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IAQDME z wywołaniem MQINQ.

QDesc (64-bajtowy łańcuch znaków)

Opis kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓	✓	✓	✓

Jest to pole, które może być używane w komentarzach opisowych. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQD.

QName (48-bajtowy łańcuch znaków)

Nazwa kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	✓

Jest to nazwa kolejki zdefiniowanej w menedżerze kolejek lokalnych. Więcej informacji na temat nazw kolejek zawiera sekcja [Reguły nazewnictwa obiektów IBM MQ](#). Wszystkie kolejki zdefiniowane w menedżerze kolejek współużytkuje tę samą przestrzeń nazw kolejki. Oznacza to, że kolejka QTLOC i kolejka QTALS nie mogą mieć takiej samej nazwy.

Aby określić wartość tego atrybutu, należy użyć selektora CAQN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

QServiceInterval (10-cyfrowa liczba całkowita ze znakiem)

Cel dla przedziału czasu usługi kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to przedział czasu usługi używany do porównania w celu wygenerowania zdarzeń OK dla okresu usługi i okresu usługi OK. Patrz atrybut *QServiceIntervalEvent*.

Wartość jest podawana w milisekundach i mieści się w zakresie od zera do 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IAQSI z wywołaniem MQINQ.

QServiceIntervalZdarzenie (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia OK dla przedziału czasu usługi lub przedziału czasu usługi OK.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

- Zdarzenie wysokiego interwału usług jest generowane, gdy sprawdzenie wskazuje, że z kolejki nie zostały pobrane żadne komunikaty co najmniej przez czas określony przez atrybut *QServiceInterval*.
- Zdarzenie Interwał usługi OK jest generowane, gdy sprawdzenie wskazuje, że komunikaty zostały pobrane z kolejki w czasie wskazanym przez atrybut *QServiceInterval*.

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Ten atrybut może mieć jedną z następujących wartości:

QSIEHI

Zdarzenia wysokiego przedziału czasu usługi kolejki są **włączone**.

- Duże zdarzenia przedziału czasu usługi kolejki są **włączone** i
- Zdarzenia OK przedziału czasu usługi kolejki są **wyłączone**.

QSIEOK

Aktywne zdarzenia przedziału czasu usługi kolejki.

- Duże zdarzenia przedziału czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK interwału usług kolejki są **włączone**.

QSIENO

Nie włączono zdarzeń odstępu czasu usługi kolejki.

- Duże zdarzenia przedziału czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK dla przedziału czasu usługi kolejki są także **wyłączone**.

W przypadku kolejek współużytkowanych wartość tego atrybutu jest ignorowana; zakłada się, że wartość QSIENO jest ustawiona.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IAQSIE z wywołaniem MQINQ.

QSGDisp (10-cyfrowa liczba całkowita ze znakiem)

Dyspozycja grupy współużytkowania kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Określa dyspozycję kolejki. Wartość ta jest jedną z następujących wartości:

QSGDQM

Dyspozycja menedżera kolejek.

Obiekt ma dyspozycję menedżera kolejek. Oznacza to, że definicja obiektu jest znana tylko z lokalnego menedżera kolejek. Definicja ta nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć obiekt o tej samej nazwie i typie, co bieżący obiekt, ale są to oddzielne obiekty i nie istnieje korelacja między nimi. Ich atrybuty nie mogą być takie same, jak w przypadku innych atrybutów.

QSGDCP

Rozporządzanie kopiami obiektów.

Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejki może mieć własną kopię tego obiektu. Początkowo wszystkie kopie mają te same atrybuty, ale za pomocą komend MQSC każda kopia może zostać zmieniona w taki sposób, aby jego atrybuty różniły się od tych z pozostałych kopii. Atrybuty kopii są resynchronizowane, gdy definicja wzorca w repozytorium współużytkowanym jest zmieniana.

QSGDSH

Rozporządzanie współużytkowane.

Obiekt ma współużytkowaną dyspozycję. Oznacza to, że we współużytkowanym repozytorium istnieje pojedyncza instancja obiektu, która jest znana wszystkim menedżerom kolejek w grupie współużytkowania kolejek. Gdy menedżer kolejek w grupie uzyskuje dostęp do obiektu, uzyskuje dostęp do pojedynczej współużytkowanej instancji obiektu.

Aby określić wartość tego atrybutu, należy użyć selektora IAQSGD przy użyciu wywołania MQINQ.

Ten atrybut jest obsługiwany tylko w systemie z/OS.

QType (10-cyfrowa liczba całkowita ze znakiem)

Typ kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	✓

Ten atrybut ma jedną z następujących wartości:

QTALS

Definicja kolejki aliasowej.

QTCLUS

Kolejka klastra.

QTLOC

Kolejka lokalna.

QTREM

Lokalna definicja kolejki zdalnej.

Aby określić wartość tego atrybutu, należy użyć selektora IAQTYP z wywołaniem MQINQ.

RemoteQMgrNazwa (48-bajtowy łańcuch znaków)

Nazwa zdalnego menedżera kolejek.

Lokalna	Model	Alias	Zdalny	Klaster
			✓	

Jest to nazwa zdalnego menedżera kolejek, w którym zdefiniowana jest kolejka *RemoteQName*. Jeśli kolejka *RemoteQName* ma wartość *QSGDisp* QSGDCP lub QSGDSH, *RemoteQMgrName* może być nazwą grupy współużytkującej kolejkę, która jest właścicielem *RemoteQName*.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, wartość *RemoteQMgrName* nie może być pusta i nie może być nazwą lokalnego menedżera kolejek. Jeśli pole *XmitQName* jest puste, jako kolejka transmisji używana jest kolejka lokalna o takiej samej nazwie, jak nazwa *RemoteQMgrName*. Jeśli nie istnieje kolejka o nazwie *RemoteQMgrName*, używana jest kolejka identyfikowana przez atrybut menedżera kolejek produktu *DefXmitQName*.

Jeśli ta definicja jest używana dla aliasu menedżera kolejek, *RemoteQMgrName* to nazwa menedżera kolejek, który jest aliasem. Może to być nazwa lokalnego menedżera kolejek. W przeciwnym razie, jeśli pole *XmitQName* jest puste w momencie otwarcia, musi istnieć kolejka lokalna o tej samej nazwie, co *RemoteQMgrName*; Ta kolejka jest używana jako kolejka transmisji.

Jeśli ta definicja jest używana na potrzeby aliasu odpowiedzi, ta nazwa jest nazwą menedżera kolejek, który ma być *MDRM*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności dla wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CARQMN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQMNM.

RemoteQName (48-bajtowy łańcuch znaków)

Nazwa kolejki zdalnej.

Lokalna	Model	Alias	Zdalny	Klaster
			✓	

Jest to nazwa kolejki, o której wiadomo, że jest ona znana w zdalnym menedżerze kolejek *RemoteQMgrName*.

Jeśli aplikacja otworzy lokalną definicję kolejki zdalnej, gdy otwarte wystąpi *RemoteQName*, nie może być puste.

Jeśli ta definicja jest używana dla definicji aliasu menedżera kolejek, wówczas gdy otwarte wystąpi *RemoteQName*, musi być puste.

Jeśli definicja jest używana na potrzeby aliasu odpowiedzi, ta nazwa jest nazwą kolejki, która ma być *MDRQ*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności dla wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CARQN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

RetentionInterval (10-cyfrowa liczba całkowita ze znakiem)

Interwał czasu przechowywania.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to czas, w którym kolejka powinna zostać zachowana. Po upływie tego czasu kolejka jest zakwalifikowana do usunięcia.

Czas jest mierzony w godzinach, licząc od daty i godziny utworzenia kolejki. Data utworzenia kolejki jest rejestrowana w *CreationDate*, a czas tworzenia kolejki jest rejestrowany w atrybucie *CreationTime*.

Te informacje są udostępniane w celu umożliwienia aplikacji porządkowej lub operatora identyfikowania i usuwania kolejek, które nie są już wymagane.

Uwaga: Menedżer kolejek nigdy nie próbuje usunąć kolejek na podstawie tego atrybutu lub aby zapobiec usuwaniu kolejek z odstępem czasu przechowywania, który nie utracił ważności. Jest to użytkownik odpowiedzialny za wykonanie wszelkich wymaganych działań, które należy podjąć.

Aby zapobiec gromadzeniu trwałych kolejek dynamicznych, należy użyć realistycznego przedziału czasu przechowywania (patrz *DefinitionType*). Jednak ten atrybut może być również używany z predefiniowanymi kolejkami.

Aby określić wartość tego atrybutu, należy użyć selektora IARINT przy użyciu wywołania MQINQ.

Zasięg (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy pozycja dla tej kolejki istnieje również w katalogu komórki.

Lokalna	Model	Alias	Zdalny	Klaster
✓		✓	✓	

Katalog komórki jest udostępniany przez usługę nazwy instalowalnej. Może to mieć jedną z następujących wartości:

SCOQM

Zasięg menedżera kolejek.

Definicja kolejki ma zasięg menedżera kolejek. Oznacza to, że definicja kolejki nie wykracza poza menedżer kolejek, który jest jego właścicielem. Aby otworzyć kolejkę dla danych wyjściowych z innego menedżera kolejek, należy podać nazwę menedżera kolejek będącego właścicielem lub inny menedżer kolejek musi mieć lokalną definicję kolejki.

ZAKCEL

Zasięg komórki.

Definicja kolejki ma zasięg komórki. Oznacza to, że definicja kolejki jest również umieszczana w katalogu komórki dostępnym dla wszystkich menedżerów kolejek w komórce. Kolejka może zostać otwarta dla danych wyjściowych dowolnego z menedżerów kolejek w komórce, podając nazwę kolejki. Nie trzeba podawać nazwy menedżera kolejek, do którego należy kolejka. Definicja kolejki nie jest jednak dostępna dla żadnego menedżera kolejek w komórce, która ma również lokalną definicję kolejki o tej nazwie, ponieważ definicja lokalna ma pierwszeństwo.

Katalog komórki jest udostępniany przez usługę nazwy instalowalnej, taką jak LDAP (Lightweight Directory Access Protocol). Należy zauważyć, że produkt IBM MQ nie obsługuje już usługi nazw DCE (Distributed Computing Environment), która była wcześniej używana do wstawiania definicji kolejek do katalogu DCE (również nie jest już obsługiwana).

Model i kolejki dynamiczne nie mogą mieć zasięgu komórki.

Ta wartość jest poprawna tylko wtedy, gdy skonfigurowana została usługa nazw obsługująca katalog komórek.

Aby określić wartość tego atrybutu, należy użyć selektora IASCOP z wywołaniem MQINQ.

Obsługa tego atrybutu podlega następującym ograniczeniom:

- W systemie IBM i atrybut jest obsługiwany, ale tylko SCOQM jest poprawny.

Współużytkowność (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy kolejka może być współużytkowana dla danych wejściowych.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Wskazuje, czy kolejka może być otwierana jednocześnie dla wielu operacji wejściowych. Może to mieć jedną z następujących wartości:

QASHR

Kolejka jest współużytkowalna.

Wielokrotne otwarcie z opcją OOINPS jest dozwolone.

QANSHR

Kolejka nie jest możliwa do współużytkowania.

Wywołanie MQOPEN z opcją OOINPS jest traktowane jako OOINPX.

Aby określić wartość tego atrybutu, należy użyć selektora IASHAR z wywołaniem MQINQ.

TriggerControl (10-cyfrowa liczba całkowita ze znakiem)

Sterowanie wyzwalaczem.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Ta opcja określa, czy komunikaty wyzwalacza są zapisywane do kolejki inicjuj, aby aplikacja została uruchomiona w celu obsługi kolejki. Jest to jedna z następujących sytuacji:

TCOFF

Komunikaty wyzwalacza nie są wymagane.

Dla tej kolejki nie ma być zapisywane komunikaty wyzwalacza. W tym przypadku wartość *TriggerType* nie ma znaczenia.

TCON

Wymagane są komunikaty wyzwalacza.

Komunikaty wyzwalacza mają być zapisywane dla tej kolejki, gdy wystąpią odpowiednie zdarzenia wyzwalające.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGC z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerData (64-bitowy łańcuch znaków)

Dane wyzwalacza.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to dane w formacie wolnym, które menedżer kolejek wstawia do komunikatu wyzwalacza, gdy komunikat przybywający do tej kolejki powoduje zapisanie komunikatu wyzwalacza w kolejce inicjuj.

Treść tych danych nie ma znaczenia dla menedżera kolejek. Ma znaczenie albo do aplikacji wyzwalacza-monitor, która przetwarza kolejkę inicjującą, albo do aplikacji, która jest uruchamiana przez monitor wyzwalacza.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora CATRGD z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET. Długość tego atrybutu jest podana przez LNTRGD.

TriggerDepth (10-cyfrowa liczba całkowita ze znakiem)

Wyzwalacz uruchamiany zapełnieniem.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej, które muszą znajdować się w kolejce, zanim zostanie zapisany komunikat wyzwalacza. Ma to zastosowanie, gdy parametr *TriggerType* jest ustawiony na wartość TTDPTH. Wartość *TriggerDepth* jest równa lub większa od jednej. Ten atrybut nie jest używany w inny sposób.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGD z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerMsgPriorytet (10-cyfrowa liczba całkowita ze znakiem)

Priorytet komunikatu progowego dla wyzwalaczy.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Jest to priorytet komunikatu, poniżej którego komunikaty nie przyczyniają się do generowania komunikatów wyzwalacza (oznacza to, że menedżer kolejek ignoruje te komunikaty podczas określania, czy komunikat wyzwalacza powinien zostać wygenerowany). *TriggerMsgPriority* może być w zakresie od zera (od najniższego) do *MaxPriority* (najwyższy; patrz [“Atrybuty dla menedżera kolejek” na stronie 1427](#)); wartość zero powoduje, że wszystkie komunikaty mogą przyczyniać się do generowania komunikatów wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGP z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerType (10-cyfrowa liczba całkowita ze znakiem)

Typ wyzwalacza.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Określa to warunki, w których komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki. Wartość ta jest jedną z następujących wartości:

TTBRAK

Brak komunikatów wyzwalacza.

Żadne komunikaty wyzwalacza nie są zapisywane w wyniku komunikatów w tej kolejce. Ma to taki sam efekt, jak ustawienie *TriggerControl* na TCOFF.

TTFRST

Wyzwalanie komunikatu, gdy głębokość kolejki trwa od 0 do 1.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce zmienia się z zakresu od 0 do 1.

TTEVRY

Wyzwalaj komunikat dla każdego komunikatu.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy w kolejce pojawia się komunikat o priorytecie *TriggerMsgPriority* lub wyższym.

TTDPTH

Komunikat wyzwalacza, gdy przekroczono próg głębokości.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce jest równa lub większa niż *TriggerDepth*. Po zapisaniu komunikatu wyzwalacza produkt *TriggerControl* jest ustawiony na wartość TCOFF, aby zapobiec dalszemu wyzwalaniu, dopóki nie zostanie jawnie ponownie włączone.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGT przy użyciu wywołania MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Składnia (10-cyfrowa liczba całkowita ze znakiem)

Użycie kolejki.

Lokalna	Model	Alias	Zdalny	Klaster
✓	✓			

Określa, dla której kolejki jest używana. Wartość ta jest jedną z następujących wartości:

USNORM

Normalne użycie.

Jest to kolejka, której normalne aplikacje używają podczas umieszczania i pobierania komunikatów. Kolejka nie jest kolejką transmisji.

USTRAN

Kolejka transmisji.

Jest to kolejka używana do przechowywania komunikatów przeznaczonych dla menedżerów kolejek zdalnych. Gdy normalna aplikacja wysyła komunikat do kolejki zdalnej, lokalny menedżer kolejek przechowuje komunikat tymczasowo w odpowiedniej kolejce transmisji w specjalnym formacie. Agent kanału komunikatów odczytuje następnie komunikat z kolejki transmisji i transportuje komunikat do zdalnego menedżera kolejek. Więcej informacji na temat kolejek transmisji zawiera sekcja [Kolejki transmisji](#).

Tylko aplikacje uprzywilejowane mogą otwierać kolejkę transmisji dla OOOOUT w celu bezpośredniego umieszczania komunikatów na niej. W tej sytuacji zwykle oczekuje się, że aplikacje narzędziowe będą mogły to zrobić. Należy zwrócić uwagę na to, że format danych komunikatu jest poprawny (patrz [“MQXQH-nagłówek kolejki transmisji”](#) na stronie 1265), w przeciwnym razie mogą wystąpić błędy podczas procesu transmisji. Kontekst nie jest przekazywany ani ustawiany, chyba że określono jedną z opcji kontekstu PM*.

Aby określić wartość tego atrybutu, należy użyć selektora IAUSAG przy użyciu wywołania MQINQ.

XmitQName (48-bajtowy łańcuch znaków)

Nazwa kolejki transmisji.

Lokalna	Model	Alias	Zdalny	Klaster
			✓	

Jeśli ten atrybut jest niepusty, gdy wystąpi otwarcie, dla kolejki zdalnej lub definicji aliasu menedżera kolejek, określa ona nazwę lokalnej kolejki transmisji, która ma być używana do przekazywania komunikatu.

Jeśli pole *XmitQName* jest puste, jako kolejka transmisji używana jest kolejka lokalna o takiej samej nazwie, jak nazwa *RemoteQMgrName*. Jeśli nie istnieje kolejka o nazwie *RemoteQMgrName*, używana jest kolejka identyfikowana przez atrybut menedżera kolejek produktu *DefXmitQName*.

Ten atrybut jest ignorowany, jeśli definicja jest używana jako alias menedżera kolejek, a *RemoteQMgrName* to nazwa lokalnego menedżera kolejek. Atrybut nie jest również brany pod uwagę, jeśli definicja jest używana jako definicja aliasu kolejki zwrotnej.

Aby określić wartość tego atrybutu, należy użyć selektora CAXQN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

Atrybuty dla list nazw

W tym temacie przedstawiono podsumowanie atrybutów, które są specyficzne dla list nazw. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy wyświetlanych atrybutów są nazwami używanymi w wywołaniach MQINQ i MQSET.

Opisy atrybutów

Obiekt listy nazw ma następujące atrybuty:

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podawana przez LNTIME.

NameCount (10-cyfrowa liczba całkowita ze znakiem)

Liczba nazw na liście nazw.

Wartość ta jest większa lub równa zero. Zdefiniowana jest następująca wartość:

NCMXNL

Maksymalna liczba nazw na liście nazw.

Aby określić wartość tego atrybutu, należy użyć selektora IANAMC przy użyciu wywołania MQINQ.

NamelistDesc (64-bajtowy łańcuch znaków)

Opis listy nazw.

Jest to pole, które może być używane w komentarzach opisowych; jego wartość jest ustalana przez proces definiowania. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole to może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CALSTD z wywołaniem MQINQ.

Długość tego atrybutu jest nadawana przez LNNLD.

NamelistName (48-bajtowy łańcuch znaków)

Nazwa listy nazw.

Jest to nazwa listy nazw, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda lista nazw ma inną nazwę niż nazwy innych list nazw należących do menedżera kolejek, ale może duplikować nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CALSTN z wywołaniem MQINQ.

Długość tego atrybutu jest nadawana przez LNNLN.

Nazwy (48-bajtowy łańcuch znaków x NameCount)

Lista nazw *NameCount*.

Każda nazwa jest nazwą obiektu, który jest zdefiniowany w lokalnym menedżerze kolejek. Więcej informacji na temat nazw obiektów zawiera sekcja [Nazewnictwo obiektów IBM MQ](#).

Aby określić wartość tego atrybutu, należy użyć selektora CANAMS z wywołaniem MQINQ.

Długość każdej nazwy na liście jest podana przez LNOBJN.

Atrybuty definicji procesów

W tym temacie podsumowano atrybuty specyficzne dla definicji procesów. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy wyświetlanych atrybutów są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli komendy MQSC są używane do definiowania, zmieniania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Szczegółowe informacje można znaleźć w sekcji [Komendy MQSC](#).

Opisy atrybutów

Obiekt definicji procesu ma następujące atrybuty:

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podawana przez LNTIME.

ApplId (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. Te informacje są przeznaczone do użycia przez aplikację monitorującą wyzwalenie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalcza.

Znaczenie *ApplId* jest określane przez aplikację wyzwalcza-monitor. Monitor wyzwalcza udostępniony przez produkt IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego.

łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora CAAPPI z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNPROA.

ApplType (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji.

Identyfikuje rodzaj programu, który ma być uruchomiony w odpowiedzi na odezwie komunikatu wyzwalacza. Te informacje są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalacza.

ApplType może mieć dowolną wartość. Dla typów standardowych można użyć następujących wartości: typy aplikacji zdefiniowane przez użytkownika są ograniczone do wartości z zakresu ATUFST przez ATULST:

równaCICS

CICS .

AT400

Aplikacja IBM i .

ATUFST

Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST

Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Aby określić wartość tego atrybutu, należy użyć selektora IAAPPT przy użyciu wywołania MQINQ.

EnvData (128-bajtowy łańcuch znaków)

Dane środowiska.

Jest to łańcuch znaków zawierający informacje dotyczące środowiska dotyczące aplikacji, która ma zostać uruchomiona. Te informacje są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj; informacje te są wysyłane do kolejki inicjuj w ramach komunikatu wyzwalacza.

Znaczenie *EnvData* jest określone przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez program IBM MQ dopisuje *EnvData* do listy parametrów przekazanej do uruchomionej aplikacji. Lista parametrów składa się ze struktury MQTMC2 , po której następują jedno puste, po którym następuje *EnvData* z usuniętymi odstępami końcowymi.

łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora CAENVD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNPROE.

ProcessDesc (64-bajtowy łańcuch znaków)

Opis procesu.

Jest to pole, które może być używane w komentarzach opisowych. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAPROD z wywołaniem MQINQ.

Długość tego atrybutu jest podana przez LNPROD.

ProcessName (48-bajtowy łańcuch znaków)

Nazwa procesu.

Jest to nazwa definicji procesu, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda definicja procesu ma nazwę różniącą się od nazw innych definicji procesów należących do menedżera kolejek. Jednak nazwa definicji procesu może być taka sama, jak nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CAPRON z wywołaniem MQINQ.

Długość tego atrybutu jest podana przez LNPRON.

UserData (128-bajtowy łańcuch znaków)

Dane użytkownika.

Jest to łańcuch znaków zawierający informacje o użytkowniku odnoszące się do aplikacji, która ma zostać uruchomiona. Informacje te są przeznaczone do użycia przez aplikację monitorującego wyzwalanie, która przetwarza komunikaty w kolejce inicjuj lub aplikacji, która jest uruchamiana przez monitor wyzwalacza. Informacje te są wysyłane do kolejki inicjuj jako część komunikatu wyzwalacza.

Znaczenie *UserData* jest określane przez aplikację wyzwalacza-monitor. Monitor wyzwalacza udostępniony przez produkt IBM MQ przekazuje produkt *UserData* do uruchomionej aplikacji jako część listy parametrów. Lista parametrów składa się ze struktury MQTMC2 (zawierającej *UserData*), po której następuje jedno puste miejsce, po którym następuje *EnvData* z usuniętą spacjami kończącymi.

Łańcuch znaków nie może zawierać żadnych wartości NULL. Jeśli jest to konieczne, jest dopełniane z prawej strony znakami pustymi.

Aby określić wartość tego atrybutu, należy użyć selektora CAUSRD z wywołaniem MQINQ. Długość tego atrybutu jest podawana przez LNPROU.

Atrybuty dla menedżera kolejek

Podsumowanie atrybutów menedżera kolejek.

Niektóre atrybuty menedżera kolejek są ustalane dla konkretnych implementacji, natomiast inne można zmienić za pomocą komendy MQSC ALTER QMGR. Atrybuty te mogą być również wyświetlane za pomocą komendy DISPLAY QMGR. Większość atrybutów menedżera kolejek można uzyskać, otwierając specjalny obiekt OTQM, a następnie za pomocą wywołania MQINQ z zwróconego uchwytu.

Poniższa tabela zawiera podsumowanie atrybutów, które są specyficzne dla menedżera kolejek. Atrybuty są opisane w kolejności alfabetycznej.

Uwaga: Nazwy atrybutów wyświetlane w tej sekcji są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli komendy MQSC są używane do definiowania, modyfikowania lub wyświetlania atrybutów, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja [Komendy skryptowe \(MQSC\)](#).

Tabela 177. Atrybuty dla menedżera kolejek	
Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
AuthorityEvent	Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane).
BridgeEvent	Określa, czy generowane są zdarzenia mostu IMS
ChannelAutoDef	Określa, czy dozwolona jest automatyczna definicja kanału.

Tabela 177. Atrybuty dla menedżera kolejek (kontynuacja)	
Atrybut	Opis
ChannelAutoDefEvent	Określa, czy generowane są zdarzenia automatycznego definiowania kanału.
ChannelAutoDefExit	Nazwa wyjścia użytkownika dla definicji kanału automatycznego
ChannelEvent	Określa, czy generowane są zdarzenia kanału
ClusterCacheTyp	Określa, czy pamięć podręczna klastra jest stała pod względem wielkości, czy też dynamicznie
DaneClusterWorkload	Dane użytkownika dla wyjścia obciążenia klastra
ClusterWorkloadWyjście	Nazwa wyjścia użytkownika dla zarządzania obciążeniem klastra
ClusterWorkloadDługość	Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra
CodedCharSetId	Identyfikator kodowanego zestawu znaków
CommandEvent	Określa, czy komunikaty zdarzeń komendy są umieszczane w kolejce.
Nazwa QName CommandInput	Nazwa kolejki wejściowej komend
CommandLevel	Poziom komendy
ConfigurationEvent	zdarzenie konfiguracji
DeadLetter-nazwa QName	Nazwa kolejki niedostarczonych komunikatów
DefClusterXmitQueueTyp	Domyślny typ kolejki transmisji klastra
Nazwa QNameDefXmit	Domyślna nazwa kolejki transmisji
DistLists	Obsługa listy dystrybucyjnej
InhibitEvent	Określa, czy mają być generowane zdarzenia zablokowanej (Inhibit Get and Inhibit Put)
LocalEvent	Określa, czy generowane są lokalne zdarzenia błędów.
LoggerEvent	Określa, czy generowane są zdarzenia dziennika odtwarzania
MaxHandles	Maksymalna liczba uchwytów
MaxMsgDługość	Maksymalna długość komunikatu w bajtach
MaxPriority	Maksymalny priorytet
MaxUncommittedkomunikatów	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy
PerformanceEvent	Określa, czy generowane są zdarzenia związane z wydajnością.
Platforma	Platforma, na której działa menedżer kolejek
PubSubTryb	Określa, czy działa mechanizm publikowania/subskrypcji, a także umieszczony w kolejce interfejs publikowania/subskrypcji.
QMgrDesc	Opis menedżera kolejek
QMgrIdentifier	Unikalny, międzyliczbowy identyfikator menedżera kolejek
QMgrName	Nazwa menedżera kolejek
RemoteEvent	Określa, czy generowane są zdalne zdarzenia błędów

Tabela 177. Atrybuty dla menedżera kolejek (kontynuacja)	
Atrybut	Opis
RepositoryName	Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium
RepositoryNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium
SSLCRLNameList ,	Nazwa obiektu listy nazw zawierającego nazwy obiektów informacji uwierzytelniającej (patrz uwaga 1)
SSLEvent	Określa, czy generowane są zdarzenia SSL
SSLKeyRepository	Położenie repozytorium kluczy SSL (patrz uwaga 1)
SSLKeyResetLiczba	Określa liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed renowacją klucza szyfrowania.
StartStopZdarzenie	Określa, czy zdarzenia uruchomienia i zatrzymania są generowane
SyncPoint	Dostępność punktu synchronizacji
RejestrowanieTraceRoute	Steruje rejestrowaniem informacji o trasie śledzenia dla komunikatów
TreeLifeCzas	Czas życia (w sekundach) tematów nieadministracyjnych
TriggerInterval	Przedział czasu komunikatu wyzwalacza
Uwagi: 1. Ten atrybut nie może zostać zapytany przy użyciu wywołania MQINQ i nie jest opisany w tej sekcji. Więcej informacji na temat tego atrybutu zawiera sekcja Change Queue Manager (Zmiana menedżera kolejek).	

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Format daty to YYYY-MM-DD, dopełniony dwoma odstępami końcowymi, aby długość 12 bajtów była długość.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT przy użyciu wywołania MQINQ. Długość tego atrybutu jest podawana przez LNTIME.

AuthorityEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane).

Atrybut AuthorityEvent musi być ustawiony na jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAAUTE przy użyciu wywołania MQINQ.

BridgeEvent (łańcuch znaków)

Ten atrybut określa, czy komunikaty zdarzeń mostu IMS są umieszczane w systemie SYSTEM.ADMIN.CHANNEL.EVENT. Jest on obsługiwany tylko w systemie z/OS.

ChannelAutoDef (dziesięciocyfrowa liczba całkowita ze znakiem)

Określa, czy dozwolona jest automatyczna definicja kanału.

Ten atrybut steruje automatyczną definicją kanałów typu CTCVR i CTSVCN. Należy pamiętać, że automatyczna definicja kanałów CTCLSD jest zawsze włączona. Może to mieć jedną z następujących wartości:

CHADDI

Automatyczne definiowanie kanału zostało wyłączone.

CHADEN

Włączono automatyczne definiowanie kanału.

Aby określić wartość tego atrybutu, należy użyć selektora IACAD przy użyciu wywołania MQINQ.

ChannelAutoDefEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia automatycznego definiowania kanału.

Ma to zastosowanie do kanałów typu CTCVR, CTSVCN i CTCLSD. Może to mieć jedną z następujących wartości:

EVRRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie i wydajność](#).

Aby określić wartość tego atrybutu, należy użyć selektora IACADE z wywołaniem MQINQ.

ChannelAutoDefExit (20-bajtowy łańcuch znaków)

Nazwa wyjścia użytkownika dla definicji kanału automatycznego.

Jeśli ta nazwa jest niepusta, a parametr *ChannelAutoDef* ma wartość CHADEN, to wyjście jest wywoływane za każdym razem, gdy menedżer kolejek ma utworzyć definicję kanału. Ma to zastosowanie do kanałów typu CTCVR, CTSVCN i CTCLSD. Następnie program obsługi wyjścia może wykonać jedną z następujących czynności:

- Zezwól na kontynuację tworzenia definicji kanału bez zmiany.
- Zmodyfikuj atrybuty definicji kanału, która została utworzona.
- Tłumić tworzenie kanału w całości.

Aby określić wartość tego atrybutu, należy użyć selektora CACADX przy użyciu wywołania MQINQ. Długość tego atrybutu jest nadawana przez LNEXN.

ChannelEvent (łańcuch znaków)

Określa, czy generowane są komunikaty o zdarzeniach kanału.

Ten atrybut określa, czy komunikaty zdarzeń kanału są umieszczane w systemie SYSTEM.ADMIN.CHANNEL.EVENT, a jeśli tak, to jaki typ komunikatów jest umieszczony w kolejce (na przykład "kanał uruchomiony", "kanał zatrzymany", "kanał nieaktywowany"). Przed implementacją tego atrybutu jedynym sposobem umieszczenia w kolejce komunikatów zdarzenia kanału było usunięcie kolejki docelowej.

Ten atrybut umożliwia również gromadzenie tylko zdarzeń mostu IMS (ponieważ można teraz wyłączyć zdarzenia kanału, nie są one umieszczane w tej samej kolejce). To samo dotyczy zdarzeń SSL, które mogą być również gromadzone bez konieczności gromadzenia zdarzeń kanału.

Ten atrybut umożliwia również gromadzenie istotnych zdarzeń (na przykład w przypadku, gdy kanały mają błędy, a nie wtedy, gdy są uruchamiane i zatrzymują się normalnie).

Wartość atrybutu ChannelEvent może być jedną z następujących wartości:

- EVREXP (generowane są tylko następujące zdarzenia kanału: RC2279, RC2283, RC2284, RC2295, RC2296).
- EVRENA (generowane są wszystkie zdarzenia kanału, to znaczy oprócz zdarzeń wygenerowanych przez EVREXP, generowane są również zdarzenia RC2282 i RC2283).
- EVRDIS (nie są generowane żadne zdarzenia kanału; jest to początkowa wartość domyślna menedżera kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora IACHNE przy użyciu wywołania MQINQ.

ClusterCacheType (32-bajtowy łańcuch znaków)

Określa, czy pamięć podręczna klastra ma stałą wielkość, czy też jest wielkością dynamiczną.

Jest to 32-bajtowy łańcuch znaków zdefiniowany przez użytkownika, który jest przekazywany do wyjścia obciążenia klastra, gdy jest on wywoływany. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWD z wywołaniem MQINQ.

ClusterWorkloadData (32-bajtowy łańcuch znaków)

Dane użytkownika dla wyjścia obciążenia klastra.

Jest to 32-bajtowy łańcuch znaków zdefiniowany przez użytkownika, który jest przekazywany do wyjścia obciążenia klastra, gdy jest on wywoływany. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWD z wywołaniem MQINQ.

ClusterWorkloadExit (20-bajtowy łańcuch znaków)

Nazwa wyjścia użytkownika dla zarządzania obciążeniem klastra.

Jeśli ta nazwa nie jest pusta, to wyjście jest wywoływane za każdym razem, gdy komunikat jest umieszczany w kolejce klastra lub przenoszony z jednej kolejki nadawczej klastra do innej. Wyjście może następnie zaakceptować instancję kolejki wybraną przez menedżer kolejek jako miejsce docelowe dla komunikatu lub wybrać inną instancję kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWX przy użyciu wywołania MQINQ. Długość tego atrybutu jest nadawana przez LNEXN.

ClusterWorkloadDługość (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra.

Jest to maksymalna długość danych komunikatu przekazywana do wyjścia obciążenia klastra. Rzeczywista długość danych przekazywanych do wyjścia to minimum:

- Długość komunikatu.
- Atrybut *MaxMsgLength* menedżera kolejek.
- Atrybut *ClusterWorkloadLength*.

Aby określić wartość tego atrybutu, należy użyć selektora IACLWL z wywołaniem MQINQ.

CodedCharSetId (dziesięciocyfrowa liczba całkowita ze znakiem)

Identyfikator kodowanego zestawu znaków.

Definiuje zestaw znaków używany przez menedżer kolejek dla wszystkich pól łańcucha znaków zdefiniowanych w interfejsie MQI, takich jak nazwy obiektów oraz data i godzina utworzenia kolejki. Zestaw znaków musi być jednym z znaków jednobajtowych dla znaków, które są poprawne w nazwach obiektów. Nie stosuje się do danych aplikacji przenoszonych w komunikacie. Wartość zależy od środowiska:

- W systemie IBM wartość ta jest ustawiana w środowisku po pierwszym utworzeniu menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IACCSI z wywołaniem MQINQ.

CommandEvent (liczba całkowita)

Określa, czy komunikaty są umieszczane w kolejce lokalnej przy wydawanych komendach.

Określa, czy komunikaty są zapisywane do nowej kolejki zdarzeń, SYSTEM.ADMIN.COMMAND.EVENT, za każdym razem, gdy komendy są wydawane. Ta funkcja jest przydatna w przypadku powiadamiania o śledzeniu komend oraz w celu diagnozowania problemów. Aby dowiedzieć się więcej o atrybucie menedżera kolejek CommandEvent, użyj nowego selektora atrybutu iacev z jedną z następujących wartości:

- Zdarzenia EVRENA-komunikaty zdarzeń są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend.
- Komunikaty zdarzeń komendy EVND są generowane i umieszczane w kolejce dla wszystkich komend zakończonych powodzeniem, innych niż komenda DISPLAY (MQSC), oraz komenda Inquire (PCF).
- Komunikaty zdarzeń komendy ECVRDIS nie są generowane lub umieszczane w kolejce (jest to początkowa wartość domyślna menedżera kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CMDEV z wywołaniem MQINQ.

CommandInputQName (48-bajtowy łańcuch znaków)

Nazwa kolejki wejściowej komend.

CommandInputNazwa QName to nazwa kolejki wejściowej komend zdefiniowana w menedżerze kolejek lokalnych. Jest to kolejka, do której użytkownicy mogą wysyłać komendy, jeśli są do tego upoważnieni. Nazwa kolejki zależy od środowiska:

- W systemie IBM inazwą kolejki jest SYSTEM.ADMIN.COMMAND.QUEUE, a tylko komendy PCF mogą być do niego wysyłane. Jednak komenda MQSC może zostać wysłana do tej kolejki, jeśli komenda MQSC jest ujęta w komendzie PCF typu CMESC. Więcej informacji na temat komendy Escape znajduje się w sekcji [Escape](#).

Aby określić wartość tego atrybutu, należy użyć selektora CACMDQ z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

CommandLevel (10-cyfrowa liczba całkowita ze znakiem)

Poziom komendy. Wskazuje to poziom komend sterujących systemem obsługiwanych przez menedżer kolejek.

Poziom jest jedną z następujących wartości:

CML700

Poziom 700 komend sterujących systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i wersja 7 wydanie 0

CML701

Poziom 701 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i wersja 7, wydanie 0, modyfikacja 1

CML71

Poziom 71 komend sterujących systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i wersja 7 wydanie 1

CML800

Poziom 800 komend sterujących systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
- 8.0

Zestaw komend sterujących systemu, które odpowiadają konkretnej wartości atrybutu *CommandLevel*, zależy od wartości atrybutu *Platform*. Oba te komendy muszą być używane do decydowania o tym, które komendy sterujące systemu są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora IACMDL przy użyciu wywołania MQINQ.

ConfigurationEvent

Określa, czy zdarzenia konfiguracji są generowane i wysyłane do systemu SYSTEM.ADMIN.CONFIG.EVENT, obiekt domyślny kolejki.

Atrybut ConfigurationEvent może mieć jedną z następujących wartości:

- EVRENA
- EVRDIS

Jeśli atrybut ConfigurationEvent jest ustawiony na wartość EVRENA, a niektóre komendy są pomyślnie wydawane za pomocą komendy runmqsc lub PCF, zdarzenia konfiguracji są generowane i wysyłane do systemu SYSTEM.ADMIN.CONFIG.EVENT. Zdarzenia dla następujących komend są wydawane, nawet jeśli komenda zmiany nie zmienia danego obiektu. Komendy, dla których są generowane i wysyłane zdarzenia konfiguracji, są następujące:

- DEFINE/ALTER AUTHINFO
- DEFINE/ALTER CHANNEL
- DEFINE/ALTER NAMELIST
- DEFINE/ALTER PROCESS
- DEFINE/ALTER QLOCAL (chyba, że jest to tymczasowa kolejka dynamiczna)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- USUŃ INFORMACJE O AUTORYZACJI
- Usun kanał
- USUŃ NAZWĘ LISTY
- Usun proces
- DELETE QLOCAL (chyba że jest to tymczasowa kolejka dynamiczna)
- DELETE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (chyba, że atrybut CONFIGEV jest wyłączony i nie został zmieniony na włączony)
- ODŚWIEŻ MENEDŻERA KOLEJEK
- Wywołanie MQSET, inne niż dla tymczasowej kolejki dynamicznej.

Zdarzenia nie są generowane (jeśli są włączone) w następujących okolicznościach:

- Wywołanie komendy lub MQSET nie powiodło się.
- Menedżer kolejek nie może umieścić komunikatu zdarzenia w kolejce zdarzeń. Komenda powinna zakończyć się pomyślnie.
- Tymczasowe kolejki dynamiczne.
- Zmiany atrybutów wewnętrznych wykonywane są bezpośrednio lub niejawnie (nie za pomocą komendy MQSET lub komendy); dotyczy to TRIGGER, CURDEPTH, IPPROCS, OPPROCS, QDPHIEV, QDPLOEV, QDPMAXEV, QSVCIEV.
- Gdy kolejka zdarzeń konfiguracji zostanie zmieniona, zostanie wygenerowany komunikat zdarzenia dla tej zmiany po zażądaniu odświeżenia.
- Grupowanie zmienia się za pomocą komend REFRESH/RESET CLUSTER i RESUME/SUSPEND QMGR.
- Tworzenie lub usuwanie menedżera kolejek.

DeadLetterQName (48-bajtowy łańcuch znaków)

Nazwa kolejki niedostarczonych komunikatów (niedostarczonych komunikatów).

Jest to nazwa kolejki zdefiniowanej w menedżerze kolejek lokalnych. Komunikaty są wysyłane do tej kolejki, jeśli nie mogą być kierowane do ich poprawnego miejsca docelowego.

Na przykład komunikaty są umieszczane w tej kolejce, gdy:

- Komunikat dociera do menedżera kolejek, który jest przeznaczony dla kolejki, która nie została jeszcze zdefiniowana w tym menedżerze kolejek.
- Komunikat dociera do menedżera kolejek, ale kolejka, do której jest przeznaczony, nie może jej odebrać, ponieważ możliwe jest:
 - Kolejka jest pełna
 - Żądania umieszczania żądań są zablokowane
 - Węzeł wysyłający nie ma uprawnień do umieszczania komunikatów w kolejce.

Aplikacje mogą również umieszczać komunikaty w kolejce niedostarczonych komunikatów.

Komunikaty raportu są traktowane w taki sam sposób, jak zwykłe komunikaty. Jeśli komunikat raportu nie może zostać dostarczony do kolejki docelowej (zwykle jest to kolejka określona w polu *MDRQ* w deskrytorze komunikatu oryginalnego komunikatu), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Uwaga: Messages that have passed their expiry time (see the *MDEXP* field described in “MQMD-deskryptor komunikatu” na stronie 1124) are **nie** transferred to this queue when they are discarded. Jednak komunikat raportu o utracie ważności (ROEXP) jest nadal generowany i wysyłany do kolejki produktu *MDRQ* , o ile jest to wymagane przez aplikację wysyłającą.

Komunikaty nie są umieszczane w kolejce niedostarczonych komunikatów (undelivered-message), gdy aplikacja, która wydała żądanie umieszczenia, została powiadomiona synchronicznie o problemie z kodem przyczyny zwróconym przez wywołanie MQPUT lub MQPUT1 (na przykład komunikat umieszczony w kolejce lokalnej, dla której żądania umieszczania są zablokowane).

Komunikaty w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) czasami zawierają dane komunikatu aplikacji z przedrostkiem struktury MQDLH. Ta struktura zawiera dodatkowe informacje, które wskazują, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Więcej informacji na temat tej struktury można znaleźć w sekcji “MQDLH -nagłówek Dead-letter” na stronie 1077 .

Ta kolejka musi być kolejką lokalną, z atrybutem *Usage* o wartości USNORM.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie jest obsługiwana przez menedżer kolejek lub nie została ona zdefiniowana, to nazwa ta jest pusta. Wszystkie menedżery kolejek produktu IBM MQ obsługują kolejkę niedostarczonych komunikatów (undelivered-message), ale domyślnie nie jest ona zdefiniowana.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie została zdefiniowana lub jest pełna lub nie do użycia z jakiegoś innego powodu, komunikat, który zostałby przesłany przez agenta kanału komunikatów, zostanie zachowany w kolejce transmisji.

Aby określić wartość tego atrybutu, należy użyć selektora CADLQ z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

DefClusterXmitQueueType (10-cyfrowa liczba całkowita ze znakiem)

Atrybut *DefClusterXmitQueueType* określa, która kolejka transmisji jest wybierana domyślnie przez kanały wysyłające klastry w celu pobierania komunikatów z, do kanałów odbierających klastry.

Wartości atrybutu **DefClusterXmitQueueType** to MQCLXQ_SCTQ lub MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Wszystkie kanały nadawcze klastra wysyłają komunikaty z kolejki
SYSTEM.CLUSTER.TRANSMIT.QUEUE. Identyfikator *correlID* komunikatów umieszczonych

w kolejce transmisji wskazuje, do którego kanału nadawczego klastra ma zostać przekazany komunikat.

Atrybut SCTQ jest ustawiany po zdefiniowaniu menedżera kolejek. To zachowanie jest niejawne w wersjach produktu IBM WebSphere MQ starszych niż 7.5. W poprzednich wersjach atrybut menedżera kolejek DefClusterXmitQueueType był nieobecny.

MQCLXQ_CHANNEL

Każdy kanał nadawczy klastra wysyła komunikaty z innej kolejki transmisji. Każda kolejka transmisji jest tworzona jako stała kolejka dynamiczna z kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Jeśli atrybut menedżera kolejek, DefClusterXmitQueue, typ, jest ustawiony na wartość CHANNEL, Konfiguracja domyślna została zmieniona w taki sposób, że kanały nadawcze klastra zostały powiązane z poszczególnymi kolejkami transmisji klastra. Kolejki transmisji to trwałe kolejki dynamiczne utworzone na podstawie kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Każda kolejka transmisji jest powiązana z jednym kanałem nadawczym klastra. Ponieważ jeden kanał nadawczy klastra obsługuje kolejkę transmisji klastra, kolejka transmisji zawiera komunikaty dla tylko jednego menedżera kolejek w jednym klastrze. Istnieje możliwość skonfigurowania klastrów w taki sposób, aby każdy menedżer kolejek w klastrze zawierał tylko jedną kolejkę klastra. W takim przypadku ruch komunikatów z menedżera kolejek do każdej kolejki klastra jest przekazywany niezależnie z komunikatów do kolejki.

Aby wysłać zapytanie o wartość, należy wywołać komendę MQINQ lub wysłać komendę Menedżer kolejek zapytania (MQCMD_INQUIRE_Q_MGR) PCF, ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE. Aby zmienić tę wartość, należy wysłać komendę PCF menedżera kolejek zmian (Change Queue Manager-MQCMD_CHANGE_Q_MGR), ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE.

Odsyłacze pokrewne

“MQINQ-zapytanie o atrybuty obiektu” na stronie 1328

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Informacje pokrewne

[Zmiana menedżera kolejek](#)

[Zapytaj menedżera kolejek](#)

DefXmitQName (48-bajtowy łańcuch znaków)

Domyślna nazwa kolejki transmisji.

Jest to nazwa kolejki transmisji używanej do przesyłania komunikatów do zdalnych menedżerów kolejek, jeśli nie ma innego wskazania, do której kolejki transmisji należy użyć.

Jeśli nie ma domyślnej kolejki transmisji, nazwa jest całkowicie pusta. Początkowa wartość tego atrybutu jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CADXQN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQN.

DistLists (10-cyfrowa liczba całkowita ze znakiem)

Obsługa listy dystrybucyjnej.

Wskazuje to, czy lokalny menedżer kolejek obsługuje listy dystrybucyjne w wywołaniach MQPUT i MQPUT1. Może to mieć jedną z następujących wartości:

DLSUPP

Obsługiwane są listy dystrybucyjne.

DLNSUP

Listy dystrybucyjne nie są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora IADIST za pomocą wywołania MQINQ.

InhibitEvent (dziesięciocyfrowa liczba całkowita ze znakiem)

Określa, czy mają być generowane zdarzenia zablokowanej (Inhibit Get and Inhibit Put).

Może to mieć jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie i wydajność](#).

Aby określić wartość tego atrybutu, należy użyć selektora IAINHE z wywołaniem MQINQ.

LocalEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są lokalne zdarzenia błędów.

Wartość ta jest jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IALCLE z wywołaniem MQINQ.

LoggerEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia programu rejestrującego odtwarzania.

Może to mieć jedną z następujących wartości:

WŁĄCZONY

Zdarzenia programu rejestrującego są generowane.

WYŁĄCZONE

Zdarzenia programu rejestrującego nie są generowane. Jest to początkowa wartość domyślna menedżerów kolejek.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie i wydajność](#).

MaxHandles (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna liczba uchwytów.

Jest to maksymalna liczba otwartych uchwytów, które mogą być używane jednocześnie przez dowolne zadanie. Każde pomyślne wywołanie MQOPEN dla jednej kolejki (lub dla obiektu, który nie jest kolejką) używa jednego uchwytu. Ten uchwyt stanie się dostępny do ponownego wykorzystania podczas zamykania obiektu. Jednak po otwarciu listy dystrybucyjnej każda kolejka na liście dystrybucyjnej przydziela osobny uchwyt, tak aby wywołanie MQOPEN używało tylu uchwytów, ile kolejek znajdujących się na liście dystrybucyjnej. Musi to być brane pod uwagę przy podejmowaniu decyzji o odpowiedniej wartości dla *MaxHandles*.

Wywołanie MQPUT1 wykonuje wywołanie MQOPEN w ramach przetwarzania. W wyniku tego wywołania MQPUT1 używa tylu uchwytów, co operacja MQOPEN, ale uchwytów są używane tylko przez czas trwania wywołania MQPUT1.

Wartość mieści się w zakresie od 1 do 999 999 999. W systemie IBM i wartością domyślną jest 256.

Aby określić wartość tego atrybutu, należy użyć selektora IAMHND przy użyciu wywołania MQINQ.

MaxMsgDługość (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna długość komunikatu w bajtach.

Jest to długość najdłuższego komunikatu *fizycznego*, który może być obsługiwany przez menedżer kolejek. Jednak ponieważ atrybut menedżera kolejek produktu *MaxMsgLength* może być ustawiony niezależnie od atrybutu kolejki produktu *MaxMsgLength*, najdłuższy komunikat fizyczny, który może zostać umieszczony w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, istnieje możliwość umieszczenia komunikatu *logicznego*, który jest dłuższy niż mniejszy z dwóch atrybutów produktu *MaxMsgLength*, ale tylko wtedy, gdy aplikacja określa flagę MFSEGA w strukturze MQMD. Jeśli ta opcja jest określona, górna granica długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zwykle ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym aplikacja jest uruchomiona, będą skutkować dolną granicą.

Dolny limit dla atrybutu *MaxMsgLength* wynosi 32 kB (32 768 bajtów). W systemie IBM maksymalna długość komunikatu wynosi 100 MB (104 857 600 bajtów).

Aby określić wartość tego atrybutu, należy użyć selektora IAMLEN przy użyciu wywołania MQINQ.

MaxPriority (10-cyfrowa liczba całkowita ze znakiem)

Maksymalny priorytet.

Jest to maksymalny priorytet komunikatu obsługiwanego przez menedżer kolejek. Priorytety są różne od zera (najniższy) do *MaxPriority* (najwyższy).

Aby określić wartość tego atrybutu, należy użyć selektora IAMPRI przy użyciu wywołania MQINQ.

MaxUncommittedMsgs (10-cyfrowa liczba całkowita ze znakiem)

Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy.

Jest to maksymalna liczba niezatwierdzonych komunikatów, które mogą istnieć w obrębie jednostki pracy. Liczba niezatwierdzonych komunikatów jest sumą następujących wartości od początku bieżącej jednostki pracy:

- Komunikaty wstawiane przez aplikację z opcją PMSYP
- Komunikaty pobrane przez aplikację z opcją GMSYP
- Komunikaty wyzwalacza i komunikaty raportu COA wygenerowane przez menedżera kolejek dla komunikatów umieszczonych za pomocą opcji PMSYP
- Komunikaty raportu COD wygenerowane przez menedżera kolejek dla komunikatów pobranych z opcją GMSYP

Następujące wartości *nie* są liczone jako komunikaty niezatwierdzone:

- Komunikaty umieszczane lub pobierane przez aplikację poza jednostką pracy
- Komunikaty wyzwalacza lub komunikaty raportu COA/COD wygenerowane przez menedżera kolejek w wyniku komunikatów umieszczanych lub pobieranych poza jednostką pracy
- Komunikaty raportu o utracie ważności wygenerowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat o utracie ważności jest określone przez GMSYP)
- Komunikaty zdarzeń wygenerowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat zdarzenia to PMSYP lub GMSYP)

Uwaga:

1. Komunikaty raportów o wyjątkach są generowane przez agenta kanału komunikatów (MCA) lub przez aplikację i są traktowane w taki sam sposób, jak zwykłe komunikaty umieszczane lub pobierane przez aplikację.
2. Jeśli komunikat lub segment jest umieszczony za pomocą opcji PMSYP, liczba niezatwierdzonych komunikatów jest zwiększana o jeden niezależnie od tego, ile fizycznych komunikatów faktycznie wynika z operacji put. (Więcej niż jeden komunikat fizyczny może spowodować, że menedżer kolejek musi podzielić się komunikatem lub segmentem).
3. Gdy lista dystrybucyjna jest umieszczana w opcji PMSYP, liczba niezatwierdzonych komunikatów jest zwiększana o jeden *dla każdego wygenerowanego komunikatu fizycznego*. Może to być tak mały, jak jeden lub tak wielki, jak liczba miejsc docelowych na liście dystrybucyjnej.

Dolny limit dla tego atrybutu wynosi 1; górny limit to 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IAMUNC z wywołaniem MQINQ.

PerformanceEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdarzenia związane z wydajnością.

Element PerformanceEvent może mieć jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IAPFME z wywołaniem MQINQ.

Platforma (10-cyfrowa liczba całkowita ze znakiem)

Platforma, na której działa menedżer kolejek.

Wskazuje to system operacyjny, w którym działa menedżer kolejek. Wartość jest następująca:

PL400

IBM i.

Tryb PubSub(10-cyfrowa liczba całkowita ze znakiem)

Określa, czy działa mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrybowania, umożliwiając aplikacjom publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez interfejs w kolejce publikowania/subskrypcji.

Może to mieć jedną z następujących wartości:

PSMCP

Mechanizm publikowania/subskrybowania działa. Dlatego możliwe jest publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego. Umieszczony w kolejce interfejs publikowania/subskrybowania nie jest uruchomiony, dlatego żaden komunikat umieszczony w kolejkach monitorowanych przez wstawiony interfejs publikowania/subskrybowania nie jest działający. To ustawienie jest używane w celu zapewnienia zgodności z produktem WebSphere Message Broker V6 lub wcześniejszymi wersjami za pomocą tego menedżera kolejek, ponieważ musi on odczytywać te same kolejki, z których normalnie jest odczytywać umieszczony w kolejce interfejs publikowania/subskrypcji.

PSMDS

Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania nie działają. Nie jest więc możliwe publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego. Wszystkie komunikaty publikowania/subskrybowania, które są umieszczane w kolejkach monitorowanych przez interfejs w kolejce publikowania/subskrypcji, nie są wykonywane.

PSMEN

Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania działają. Dlatego możliwe jest publikowanie/subskrybowanie za pomocą aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez interfejs w kolejce publikowania/subskrypcji. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora PSMODE z wywołaniem MQINQ.

QMGrDesc (łańcuch znakowy 64-bajtowy)

Opis menedżera kolejek.

Jest to pole, które może być używane w komentarzach opisowych. Treść tego pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole to może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek produktu *CodedCharSetId*), znaki te mogą zostać przetłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

W systemie IBM i wartość domyślna to odstępy.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMD z wywołaniem MQINQ. Długość tego atrybutu jest podawana przez LNQMD.

QMGrIdentifier (48-bajtowy łańcuch znaków)

Unikalny, międzyliczbowy identyfikator menedżera kolejek.

Jest to unikalna nazwa, która jest generowana jako unikalna dla menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMID przy użyciu wywołania MQINQ. Długość tego atrybutu jest podawana przez LNQMID.

QMGrName (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek.

Jest to nazwa lokalnego menedżera kolejek, tj. nazwa menedżera kolejek, z którym połączona jest aplikacja.

Pierwsze 12 znaków nazwy jest używane do konstruowania unikalnego identyfikatora komunikatu (patrz pole *MDMID* opisane w sekcji “MQMD-deskryptor komunikatu” na stronie 1124). W związku z tym menedżery kolejek, które mogą wzajemnie się komunikować, muszą mieć nazwy różniące się od pierwszych 12 znaków, aby identyfikatory komunikatów były unikalne w sieci menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQMN.

RemoteEvent (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy generowane są zdalne zdarzenia błędów.

Wartość ta jest jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IARMTE przy użyciu wywołania MQINQ.

RepositoryName (48-bajtowy łańcuch znaków)

Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium.

Jest to nazwa klastra, dla którego ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę dla więcej niż jednego klastra, *RepositoryNameList* określa nazwę obiektu listy nazw, która identyfikuje klastry, a *RepositoryName* jest pusta. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNameList* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CARPN z wywołaniem MQINQ. Długość tego atrybutu jest podana przez LNQMN.

RepositoryNameList (48-bajtowy łańcuch znaków)

Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium.

Jest to nazwa obiektu listy nazw, który zawiera nazwy klastrów, dla których ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę tylko dla jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć opcji

RepositoryName do określenia nazwy klastra, w którym to przypadku pole *RepositoryNameList* jest puste. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNameList* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CARPNL przy użyciu wywołania MQINQ. Długość tego atrybutu jest nadawana przez LNNLN.

SSLEvent (łańcuch znaków)

Określa, czy generowane są zdarzenia SSL.

Wartość ta jest jedną z następujących wartości:

- EVRENA (MQINQ/PCF/config event) ENABLED (MQSC): generowane są zdarzenia SSL (to znaczy, że generowane jest zdarzenie RC2371).
- EVRDIS (MQINQ/PCF/config event) DISABLED (MQSC): zdarzenia SSL nie są generowane. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IASSLE z wywołaniem MQINQ.

SSLKeyResetLiczb (liczba całkowita)

Określa łączną liczbę niezaszyfrowanych bajtów, które są wysyłane i odbierane w ramach konwersacji SSL, zanim klucz tajny zostanie renegotjowany. Liczba bajtów obejmuje informacje sterujące wysyłane przez agenta kanału komunikatów (MCA).

Ta wartość jest używana tylko przez MCAs kanału SSL, które inicjują komunikację z tego menedżera kolejek (to znaczy kanał nadawczy MCA w parowaniu nadawcy i kanału odbiorczego).

Jeśli wartość tego atrybutu jest większa niż 0, a pulsy kanału są włączone dla kanału, klucz tajny jest również renegotjowany przed wysłaniem lub odebraniem danych po pulsie kanału. Liczba bajtów do czasu zresetowania następnej operacji renegotjacji klucza tajnego po każdej pomyślnej renegotjacji.

Wartość może być z zakresu od 0 do 999 999 999. Wartość 0 dla tego atrybutu wskazuje, że klucz tajny nigdy nie jest renegotjowany. Jeśli zostanie określona wartość resetowania klucza tajnego SSL/TLS w zakresie od 1 bajtu do 32 kB, kanały SSL/TLS będą używać klucza tajnego resetowania klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernych resetów klucza, które nastąpiłyby w przypadku małych wartości resetowania klucza tajnego SSL/TLS.

Jeśli serwer SSL jest menedżerem kolejek produktu IBM MQ , a resetowanie klucza tajnego i puls kanałów są włączone, ponowne negocjowanie odbywa się natychmiast po każdym pulsie kanału.

Aby określić wartość tego atrybutu, należy użyć selektora IASSRC przy użyciu wywołania MQINQ.

Zdarzenie StartStop(10-cyfrowa liczba całkowita ze znakiem)

Określa, czy zdarzenia uruchomienia i zatrzymania są generowane.

Ten atrybut może mieć jedną z następujących wartości:

EVRDIS

Raportowanie zdarzeń jest wyłączone.

EVRENA

Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja [Monitorowanie zdarzeń](#).

Aby określić wartość tego atrybutu, należy użyć selektora IASSE przy użyciu wywołania MQINQ.

SyncPoint (10-cyfrowa liczba całkowita ze znakiem)

Dostępność punktu synchronizacji.

Wskazuje to, czy lokalny menedżer kolejek obsługuje jednostki pracy i syncwskazujących wywołania MQGET, MQPUT i MQPUT1 .

SPAVL

Jednostki pracy i metody synchronizacji dostępne.

SPNAVL

Jednostki pracy i syncwskazujący nie są dostępne.

Aby określić wartość tego atrybutu, należy użyć selektora IASYNCR przy użyciu wywołania MQINQ.

TraceRouteRejestrowanie (10-cyfrowa liczba całkowita ze znakiem)

Ta opcja określa, czy informacje o komunikatach są rejestrowane podczas przepływu przez menedżer kolejek.

Wartość ta jest jedną z następujących wartości:

- RECD: brak możliwości dopisania do śledzenia komunikatów trasy
- RECDQ: komunikaty są umieszczane w stałej nazwanej kolejce
- RECDM: określanie użycia komunikatu (jest to początkowe ustawienie domyślne)

Aby zapobiec pozostaniu komunikatu trasy śledzenia w systemie, należy ustawić dla niej wartość utraty ważności większą niż zero, a następnie określić opcję raportu RODISC. Aby zapobiec pozostaniu w systemie komunikatów raportu lub odpowiedzi, należy ustawić opcję raportu ROPDAE. Więcej informacji na ten temat zawiera sekcja [“Opcje raportów i flagi komunikatów”](#) na stronie 1462.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGI z wywołaniem MQINQ.

Czas TreeLife(10-cyfrowa liczba całkowita ze znakiem)

Czas życia (w sekundach) tematów nieadministracyjnych.

Tematy nieadministracyjne to te, które są tworzone, gdy aplikacja publikuje lub subskrybuje łańcuch tematu, który nie istnieje jako węzeł administracyjny. Kiedy w danym węźle nieadministracyjnym nie ma już żadnych aktywnych subskrypcji, ten parametr określa czas, przez jaki menedżer kolejek będzie oczekiwać przed usunięciem tego węzła. Tylko te tematy nieadministracyjne, które są używane w ramach trwałej subskrypcji, przetrwają przetwarzanie wtórne menedżera kolejek.

Podaj wartość z zakresu od 0 do 604 000. Wartość 0 oznacza, że tematy nieadministracyjne nie są usuwane przez menedżer kolejek. Domyślna wartość początkowa menedżera kolejek to 1800.

Aby określić wartość tego atrybutu, należy użyć selektora IATRLFT przy użyciu wywołania MQINQ.

TriggerInterval (10-cyfrowa liczba całkowita ze znakiem)

Przedział czasu komunikatu wyzwalacza.

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to znaczenie tylko wtedy, gdy *TriggerType* to TTFIRST. W takim przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawia się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach dodatkowy komunikat wyzwalający może zostać wygenerowany z wyzwaniem TTFIRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekundy.

Więcej informacji na temat wyzwiania zawiera sekcja [Wyzwalanie kanałów](#).

Wartość mieści się w zakresie od zera do 999 999 999. Wartość domyślna to 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGI z wywołaniem MQINQ.

Aplikacje

W tej sekcji opisano przykładowe programy dostarczone wraz z programem IBM MQ for IBM i dla języka RPG. Dowiedz się również, jak budować aplikacje wykonywalne z napisanych przez siebie programów.

Budowanie aplikacji

Publikacje dotyczące produktu IBM i opisują sposób budowania aplikacji wykonywalnych z napisanych przez użytkownika programów. W tym temacie opisano dodatkowe zadania i zmiany w standardowych

zadaniach, które należy wykonać podczas budowania aplikacji produktu IBM MQ for IBM i w celu uruchomienia w ramach produktu IBM i.

Oprócz kodowania wywołań MQI w kodzie źródłowym, należy dodać odpowiednie instrukcje języka, aby dołączyć pliki kopii produktu IBM MQ for IBM i do języka RPG. Użytkownik powinien zapoznać się z treścią tych plików; ich nazwy oraz krótki opis ich zawartości podany jest w poniższym tekście.

Pliki kopii produktu IBM MQ

Program IBM MQ for IBM i udostępnia pliki kopii, które ułatwiają pisanie aplikacji w języku programowania RPG. Są one odpowiednie do użycia razem z zestawem narzędzi programistycznych produktu WebSphere (5722 WDS) ILE RPG 4 Compiler.

Pliki kopii, które program IBM MQ for IBM i udostępnia do obsługi zapisu wyjść kanałów, są opisane w sekcji [Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów](#).

Nazwy plików kopii produktu IBM MQ for IBM i dla języka RPG mają przedrostek CMQ. Mają przyrostek G lub H. Istnieją osobne pliki kopii zawierające nazwane stałe i jeden plik dla każdej z tych struktur. Pliki kopii są wymienione w sekcji ["Uwagi dotyczące języka"](#) na stronie 1022.

Uwaga: W środowisku ILE RPG/400 są one dostarczane jako elementy zbioru QRPGLSRC w bibliotece QMQM.

Deklaracje struktury nie zawierają instrukcji DS. Dzięki temu aplikacja może deklarować strukturę danych (lub strukturę danych o wielu zdarzeniach), kodując instrukcję DS i używając instrukcji /COPY do skopiowania w pozostałej części deklaracji:

W przypadku ILE RPG/400 instrukcja jest następująca:

```
D*..1.....2.....3.....4.....5.....6.....7
D* Declare an MQMD data structure
D MQMD          DS
D/COPY CMQMDG
```

Przygotowanie programów do uruchomienia

Aby utworzyć wykonywalną aplikację IBM MQ for IBM i, należy skompilować kod źródłowy, który został napisany.

Aby to zrobić w środowisku ILE RPG/400, można użyć typowych komend IBM i, CRTRPGMOD i CRTPGM.

Po utworzeniu *MODULE należy podać BNDSRVPGM (QMQM/LIBMQM) w komendzie CRTPGM. Obejmuje to różne procedury IBM MQ w programie.

Podczas wykonywania kompilacji należy upewnić się, że biblioteka zawierająca pliki kopii (QMQM) znajduje się na liście bibliotek.

Więcej informacji na temat zagadnień związanych z programowaniem, w tym trybów klienta, zawiera sekcja ["Uwagi dotyczące języka"](#) na stronie 1022.

Interfejsy do zewnętrznego menedżera synchronizacji punktu synchronizacji produktu IBM i

Produkt IBM MQ for IBM i używa rodzimej kontroli transakcji produktu IBM i jako zewnętrznego koordynatora punktu synchronizacji.

Więcej informacji na temat możliwości kontroli transakcji w produkcie IBM i zawiera publikacja *IBM i Programming: Backup and Recovery Guide*.

Aby uruchomić narzędzia kontroli transakcji produktu IBM i, należy użyć komendy systemowej STRCMTCTL. Aby zakończyć kontrolę transakcji, należy użyć komendy systemowej ENDCMTCTL.

Uwaga: Wartością domyślną pola *Zasięg definicji transakcji* jest *ACTGRP. Wartość ta musi być zdefiniowana jako *JOB dla IBM MQ dla IBM i. Na przykład:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

W przypadku wywołania MQPUT, MQPUT1 lub MQGET, określając PMSYP lub GMSYP, po uruchomieniu kontroli transakcji, produkt IBM MQ for IBM i dodaje siebie jako zasób zaangażowania API do definicji kontroli transakcji. Zwykle jest to pierwsze tego typu wywołanie w zadaniu. Chociaż istnieją jakiekolwiek zasoby zatwierdzania transakcji zarejestrowane w ramach określonej definicji kontroli transakcji, nie można zakończyć kontroli transakcji dla tej definicji.

Produkt IBM MQ for IBM i usuwa jego rejestrację jako zasób transakcji API po rozłączeniu z menedżerem kolejek pod warunkiem, że w bieżącej jednostce pracy nie są wykonywane żadne oczekujące operacje MQI.

W przypadku rozłączenia się z menedżerem kolejek, gdy w bieżącej jednostce pracy istnieją oczekujące operacje MQPUT, MQPUT1 lub MQGET, produkt IBM MQ for IBM i pozostaje zarejestrowany jako zasób zatwierdzania transakcji API, dzięki czemu zostanie powiadomiony o następnym zatwierdzeniu lub wycofaniu. Po osiągnięciu następnego punktu synchronizacji program IBM MQ zatwierdza lub wycofuje zmiany zgodnie z wymaganiami. Aplikacja umożliwia rozłączenie i ponowne nawiązanie połączenia z menedżerem kolejek podczas aktywnej jednostki pracy i wykonanie kolejnych operacji MQGET i MQPUT w tej samej jednostce pracy (to jest oczekiwanie na rozłączenie).

Jeśli zostanie podjęta próba wydania komendy systemowej ENDCMTCTL dla tej definicji kontroli transakcji, zostanie wyświetlony komunikat CPF8355 wskazujący, że oczekujące zmiany były aktywne. Ten komunikat pojawia się również w protokole zadania po zakończeniu zadania. Aby tego uniknąć, należy się upewnić, że wszystkie oczekujące operacje programu IBM MQ zostały zatwierdzone lub wycofane, a następnie należy odłączyć się od menedżera kolejek. Dlatego użycie komend COMMIT lub ROLLBACK przed zakończeniem działania komendy ENDCMTCTL powinno umożliwić pomyślne zakończenie kontroli transakcji.

Gdy program IBM i kontroli transakcji jest używany jako zewnętrzny koordynator punktu synchronizacji, wywołania MQCMIT, MQBACK i MQBEGIN mogą nie zostać wydane. Wywołania tych funkcji kończą się niepowodzeniem z kodem przyczyny RC2012.

Aby zatwierdzić lub wycofać zmiany (to jest, aby wycofać) jednostkę pracy, należy użyć jednego z języków programowania, który obsługuje kontrolę transakcji. Na przykład:

- Komendy CL: COMMIT i ROLLBACK
- Funkcje programowania ILE C: _Rcommit i _Rollback
- RPG/400: ZATWIERDŹ i ROLBK
- COBOL/400: COMMIT i ROLLBACK

Punkty synchronizacji w produkcji CICS dla aplikacji IBM i

Produkt IBM MQ for IBM i uczestniczy w jednostkach pracy z produktem CICS. Interfejsu MQI można używać w aplikacji CICS w celu umieszczania i pobierania komunikatów w bieżącej jednostce pracy.

Za pomocą komendy EXEC CICS SYNCPOINT można ustanowić punkt synchronizacji, który zawiera operacje IBM MQ for IBM i. Aby wycofać wszystkie zmiany w górę do poprzedniego punktu synchronizacji, można użyć komendy EXEC CICS SYNCPOINT ROLLBACK.

W przypadku użycia komendy MQPUT, MQPUT1 lub MQGET z opcją PMSYP lub GMSYP w aplikacji CICS nie można wylogować się z programu CICS, dopóki produkt IBM MQ for IBM i nie usunie swojej rejestracji jako zasobu zatwierdzania interfejsu API. Dlatego przed rozłączeniem się z menedżerem kolejek należy zatwierdzić lub wycofać wszystkie oczekujące operacje put lub get. Pozwoli to wylogować się z programu CICS.

Przykładowe programy w systemie IBM i

W tej sekcji opisano przykładowe programy dostarczane wraz z programem IBM MQ for IBM i dla języka RPG. Przykłady demonstrują typowe zastosowania interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Przykłady nie mają na celu demonstrować ogólnych technik programowania, dlatego niektóre sprawdzanie błędów, które może być uwzględnione w programie produkcyjnym, zostało pominięte. Te przykłady są jednak odpowiednie do użycia jako podstawa dla własnych programów kolejkowania komunikatów.

Kod źródłowy dla wszystkich przykładów jest dostarczany wraz z produktem; źródło to zawiera komentarze, które wyjaśniają techniki kolejkowania komunikatów demonstrowane w programach.

Istnieje jeden zestaw przykładowych programów ILE:

1. Programy korzystające z wywołań prototypowych do interfejsu MQI (połączenia statyczne)

Źródło istnieje w QMQMSAMP/QRPGLESRC. Członkowie mają nazwę AMQ3xxx4, gdzie xxx wskazuje funkcję przykładową. Elementy kopii istnieją w QMQM/QRPGLESRC. Każda nazwa elementu ma przyrostek G lub H.

Program [Tabela 178](#) na stronie 1444 zawiera pełną listę przykładowych programów dostarczanych wraz z produktem IBM MQ for IBM i, a także nazwy programów w każdym obsługiwany języku programowania. Należy zauważyć, że wszystkie nazwy rozpoczynają się od przedrostka AMQ, a czwarty znak w nazwie wskazuje język programowania.

Tabela 178. Nazwy przykładowych programów	
	RPG (ILE)
Umieść próbki	AMQ3PUT4
Przeglądanie przykładów	AMQ3GBR4
Pobierz przykłady	AMQ3GET4
Przykłady żądań	AMQ3REQ4
Przykłady echa	AMQ3ECH4
Sprawdź przykłady	AMQ3INQ4
Ustaw przykłady	AMQ3SET4
Przykład monitora wyzwalacza	AMQ3TRG4
Przykład serwera wyzwalacza	AMQ3SRV4

Oprócz tych opcji przykładowa opcja IBM MQ for IBM i zawiera przykładowy plik danych AMQSDATA, który może być używany jako dane wejściowe dla niektórych programów przykładowych i przykładowych programów CL, które demonstrują zadania administracyjne. Przykłady języka CL są opisane w sekcji [Administrowanie produktem IBM i](#). Można użyć przykładowego programu CL, aby utworzyć kolejki, które będą używane z przykładowymi programami opisanymi w tym temacie.

Informacje na temat uruchamiania przykładowych programów zawiera sekcja [“Przygotowywanie i uruchamianie przykładowych programów w systemie IBM i”](#) na stronie 1445.

Funkcje demonstrować w przykładowych programach w systemie IBM i

Tabela, w której przedstawiono techniki demonstrowane przez programy przykładowe produktu IBM MQ for IBM i.

Niektóre techniki występują w więcej niż jednym programie przykładowym, ale w tabeli znajduje się tylko jeden program. Wszystkie przykładowe kolejki są otwierane i zamykane przy użyciu wywołań MQOPEN i MQCLOSE, dlatego te techniki nie są wymienione oddzielnie w tabeli.

<i>Tabela 179. Przykładowe programy demonstruje użycie interfejsu MQI</i>	
Technika	RPG (ILE)
Korzystanie z wywołań MQCONN i MQDISC	AMQ3ECH4 lub AMQ3INQ4
Niejawnie łączenie i rozłączanie	AMQ3PUT4
Umieszczanie komunikatów przy użyciu wywołania MQPUT	AMQ3PUT4
Umieszczanie pojedynczego komunikatu przy użyciu wywołania MQPUT1	AMQ3ECH4 lub AMQ3INQ4
Odpowiadanie na komunikat żądania	AMQ3INQ4
Pobieranie komunikatów (bez oczekiwania)	AMQ3GBR4
Pobieranie komunikatów (czekaj z limitem czasu)	AMQ3GET4
Pobieranie komunikatów (z konwersją danych)	AMQ3ECH4
Przeglądanie kolejki	AMQ3GBR4
Korzystanie z współużytkowanej kolejki wejściowej	AMQ3INQ4
Korzystanie z wyłącznej kolejki wejściowej	AMQ3REQ4
Korzystanie z wywołania MQINQ	AMQ3INQ4
Korzystanie z wywołania MQSET	AMQ3SET4
Korzystanie z kolejki odpowiedzi	AMQ3REQ4
Żądanie komunikatów o wyjątkach	AMQ3REQ4
Akceptowanie obciętej wiadomości	AMQ3GBR4
Korzystanie z przetłumaczanej nazwy kolejki	AMQ3GBR4
Przetwarzanie wyzwalacza	AMQ3SRV4 lub AMQ3TRG4

Uwaga: Wszystkie przykładowe programy generują plik buforowy zawierający wyniki przetwarzania.

Przygotowywanie i uruchamianie przykładowych programów w systemie IBM i

Aby można było uruchomić programy przykładowe produktu IBM MQ for IBM i, należy je skompilować tak, jak inne aplikacje produktu IBM MQ for IBM i. Aby to zrobić, można użyć komend IBM i CRTPRPGMOD i CRTPGM.

Podczas tworzenia programów AMQ3xxx4 należy podać wartość BNDSRVPGM (QMQM/LIBMQM) w komendzie CRTPGM. W ten sposób w programie znajdują się różne procedury IBM MQ.

Programy przykładowe są udostępniane w bibliotece QMQMSAMP jako członkowie QRPGLSRC. Używają one plików kopii udostępnionych w bibliotece QMQM, dlatego należy się upewnić, że biblioteka znajduje się na liście bibliotek podczas ich kompilowania. Kompilator języka RPG podaje komunikaty informacyjne, ponieważ w próbkach nie są używane wiele zmiennych zadeklarowanych w plikach kopii.

Uruchamianie przykładowych programów

Podczas uruchamiania przykładów można użyć własnych kolejek lub można skompilować i uruchomić komendę AMQSAMP4 w celu utworzenia niektórych kolejek przykładowych. Źródło tego programu jest dostarczane w zbiorze QCLSRC w bibliotece QMQMSAMP. Można go skompilować za pomocą komendy CRTCLPGM.

Aby wywołać jeden z przykładowych programów, należy użyć następującej komendy:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` muszą mieć długość 48 znaków, co oznacza, że dopełnianie `Queue_Name` i `Queue_Manager_Name` jest wymagane z wymaganą liczbą odstępów.

W przypadku programów przykładowych `Inquire` i `Set` przykładowe definicje utworzone przez komendę `AMQSAMP4` powodują wyzwolenie wersji C tych przykładów. Aby wyzwolić wersje RPG, należy zmienić definicje procesów `SYSTEM.SAMPLE.ECHOPROCESS` i `SYSTEM.SAMPLE.INQPROCESS` i `SYSTEM.SAMPLE.SETPROCESS`. Można użyć komendy `CHGMQMPRC` (opisanej w sekcji [Change MQ Process \(CHGMQMPRC\)](#)) w tym celu lub w celu edycji i uruchomienia produktu `AMQSAMP4` z alternatywną definicją.

Przykładowy program Put w systemie IBM i

Przykładowy program `Put`, `AMQ3PUT4`, umieszcza komunikaty w kolejce przy użyciu wywołania `MQPUT`.

Aby uruchomić program, należy wywołać program i podać nazwę kolejki docelowej jako parametr programu. Program umieszcza zestaw stałych komunikatów w kolejce; komunikaty te są pobierane z bloku danych na końcu kodu źródłowego programu. Przykładowy program wstawiany to `AMQ3PUT4` w bibliotecę `QMQMSAMP`.

W tym przykładowym programie komenda jest następująca:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` *muszą* mieć 48 znaków długości, co jest osiąganego przez dopełnienie `Queue_Name` i `Queue_Manager_Name` z wymaganą liczbą odstępów.

Projekt przykładowego programu "Put"

Program korzysta z wywołania `MQOPEN` z opcją `OOOUT`, aby otworzyć kolejkę docelową w celu umieszczania komunikatów. Wyniki są wyprowadzane do zbioru buforowego. Jeśli nie jest możliwe otwarcie kolejki, program zapisuje komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie `MQOPEN`. Aby program był prosty, w tym i w kolejnych wywołaniach `MQI` program używa wartości domyślnych dla wielu opcji.

Dla każdego wiersza danych zawartego w kodzie źródłowym program odczytuje tekst do buforu i korzysta z wywołania `MQPUT` w celu utworzenia komunikatu datagramu zawierającego tekst tego wiersza. Program będzie kontynuowany aż do momentu, gdy dojdzie do końca danych wejściowych lub wywołanie `MQPUT` nie powiedzie się. Jeśli program dociera do końca danych wejściowych, zamyka kolejkę za pomocą wywołania `MQCLOSE`.

Przykładowy program Przeglądaj w systemie IBM i

Przykładowy program `AMQ3GBR4` przegląda komunikaty w kolejce przy użyciu wywołania `MQGET`.

Program pobiera kopie wszystkich komunikatów znajdujących się w kolejce, która została określona podczas wywoływania programu; komunikaty pozostają w kolejce. Można użyć podanej kolejki `SYSTEM.SAMPLE.LOCAL`; najpierw uruchom przykładowy program umieszczania w celu umieszczenia niektórych komunikatów w kolejce. W tym celu można użyć kolejki `SYSTEM.SAMPLE.ALIAS`, która jest nazwą aliasu dla tej samej kolejki lokalnej. Program będzie kontynuowany do momentu osiągnięcia końca kolejki lub wywołanie `MQI` nie powiedzie się.

Przykładem komendy do wywołania programu RPG jest:

```
CALL PGM(QMQMSAMP/AMQ3GBR4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` *muszą* mieć 48 znaków długości, co jest osiąganego przez dopełnienie `Queue_Name` i `Queue_Manager_Name` z wymaganą liczbą odstępów. Dlatego też, jeśli używany jest plik `SYSTEM.SAMPLE.LOCAL` jako kolejka docelowa, wymagane będzie 29 pustych znaków.

Projekt przykładowego programu przeglądania

Program otwiera kolejkę docelową przy użyciu wywołania MQOPEN z opcją OOBRW. Jeśli nie może otworzyć kolejki, program zapisze komunikat o błędzie do jego pliku buforowego, zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego komunikatu w kolejce program korzysta z wywołania MQGET w celu skopiowania komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET używa następujących opcji:

GMBRWN

Po wywołaniu MQOPEN kursor przeglądania jest ustawiony logicznie przed pierwszym komunikatem w kolejce, dlatego ta opcja powoduje, że komunikat *pierwszy* jest zwracany w momencie, gdy wywołanie jest wykonywane po raz pierwszy.

GMNWT

Program nie czeka, jeśli w kolejce nie ma żadnych komunikatów.

GMATM

Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, w programie zostanie wyświetlony obcięty komunikat wraz z ostrzeżeniem, że komunikat został obcięty.

Program demonstruje sposób, w jaki należy usunąć pola *MDMID* i *MDCID* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który pobiera. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Program jest kontynuowany do końca kolejki. W tym miejscu wywołanie MQGET zwraca kod przyczyny RC2033 (brak dostępnego komunikatu), a program wyświetli komunikat ostrzegawczy. Jeśli wywołanie MQGET nie powiedzie się, program zapisze komunikat o błędzie zawierający kod przyczyny w jego pliku buforowo-buforowy.

Następnie program zamknie kolejkę przy użyciu wywołania MQCLOSE.

Program do pobierania próbek w systemie IBM i

Program pobierania próbek, AMQ3GET4, pobiera komunikaty z kolejki za pomocą wywołania MQGET.

Gdy wywoływany jest program, usuwa on komunikaty z podanej kolejki. Można użyć podanej kolejki SYSTEM.SAMPLE.LOCAL; najpierw uruchom przykładowy program umieszczania w celu umieszczenia niektórych komunikatów w kolejce. Można użyć pliku SYSTEM.SAMPLE.ALIAS, która jest nazwą aliasu dla tej samej kolejki lokalnej. Program jest kontynuowany do momentu, gdy kolejka nie będzie pusta lub wywołanie MQI nie powiedzie się.

Przykładem komendy do wywołania programu RPG jest:

```
CALL PGM(QMQMSAMP/AMQ3GET4) PARM('Queue_Name','Queue_Manager_Name')
```

gdzie Queue_Name i Queue_Manager_Name *muszą* mieć 48 znaków długości, co jest osiąganego przez dopełnienie Queue_Name i Queue_Manager_Name z wymaganą liczbą odstępów. Dlatego też, jeśli używany jest plik SYSTEM.SAMPLE.LOCAL jako kolejka docelowa, wymagane będzie 29 pustych znaków.

Projekt przykładowego programu Get

Program otwiera kolejkę docelową w celu uzyskania komunikatów. Program ten używa wywołania MQOPEN z opcją OOINPQ. Jeśli nie jest możliwe otwarcie kolejki, program zapisuje komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN w jego pliku buforowy.

Dla każdego komunikatu w kolejce program korzysta z wywołania MQGET w celu usunięcia komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET korzysta z opcji GMWT, określając przedział czasu oczekiwania (*GMWT*) na 15 sekund, przez co program oczekuje na ten okres, jeśli w kolejce nie ma komunikatu. Jeśli przed upływem tego odstępu czasu nie zostanie

wyświetlony żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2033 (brak dostępnego komunikatu).

Program demonstruje sposób, w jaki należy usunąć pola *MDMID* i *MDCID* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który pobiera. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, wywołanie nie powiedzie się, a program zostanie zatrzymany.

Program będzie kontynuowany do czasu, aż wywołanie MQGET zwróci kod przyczyny RC2033 (brak dostępnego komunikatu) lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, w programie zostanie wyświetlony komunikat o błędzie zawierający kod przyczyny.

Następnie program zamknie kolejkę przy użyciu wywołania MQCLOSE.

Przykładowy program żądania w systemie IBM i

Przykładowy program żądania AMQ3REQ4 demonstruje przetwarzanie klient/serwer. Przykład to klient, który umieszcza komunikaty żądań w kolejce, która jest przetwarzana przez program serwera. Oczekuje on na umieszczenie komunikatu odpowiedzi w kolejce zwrotnej przez program serwera.

Przykład żądania umieszcza serię komunikatów żądań w kolejce przy użyciu wywołania MQPUT. Te komunikaty określają parametr SYSTEM.SAMPLE.REPLY jako kolejkę odpowiedzi. Program oczekuje na komunikaty odpowiedzi, a następnie wyświetla je. Odpowiedzi są wysyłane tylko wtedy, gdy kolejka docelowa (o której będziemy wywoływać *kolejkę serwera*) jest przetwarzany przez aplikację serwera lub jeśli aplikacja jest wyzwolana w tym celu (programy Inquire i Set są przeznaczone do wyzwolenia). Próbkę czeka 5 minut na pierwszą odpowiedź na przyjazd (w celu umożliwienia wyzwolenia aplikacji serwera) i 15 sekund na kolejne odpowiedzi, ale może zakończyć się bez uzyskania odpowiedzi.

Aby uruchomić program, należy wywołać program i podać nazwę kolejki docelowej jako parametr programu. Program umieszcza zestaw statych komunikatów w kolejce; komunikaty te są pobierane z bloku danych na końcu kodu źródłowego programu.

Projekt przykładowego programu żądania

Program otwiera kolejkę serwera, dzięki czemu może on umieszczać komunikaty. Używa on wywołania MQOPEN z opcją OOOUT. Jeśli nie można otworzyć kolejki, w programie wyświetlany jest komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Następnie program otwiera kolejkę zwrotną o nazwie SYSTEM.SAMPLE.REPLY, aby można było uzyskać komunikaty odpowiedzi. W tym celu program korzysta z wywołania MQOPEN z opcją OOINPX. Jeśli nie można otworzyć kolejki, w programie wyświetlany jest komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i korzysta z wywołania MQPUT w celu utworzenia komunikatu żądania zawierającego tekst tego wiersza. W tym wywołaniu program korzysta z opcji raportu ROEXCD, aby zażądać, aby wszystkie komunikaty raportu wysłane na temat komunikatu żądania zawierały pierwsze 100 bajtów danych komunikatu. Program będzie kontynuowany aż do momentu, gdy dojdzie do końca danych wejściowych lub wywołanie MQPUT nie powiedzie się.

Następnie program korzysta z wywołania MQGET w celu usunięcia komunikatów odpowiedzi z kolejki i wyświetla dane zawarte w odpowiedziach. Wywołanie MQGET używa opcji GMWT, określając przedział czasu oczekiwania (*GMWI*) na 5 minut dla pierwszej odpowiedzi (w celu umożliwienia wyzwolenia aplikacji serwera) i 15 sekund na kolejne odpowiedzi. Program czeka na te okresy, jeśli w kolejce nie ma żadnego komunikatu. Jeśli przed upływem tego odstępu czasu nie zostanie wyświetlony żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2033 (brak dostępnego komunikatu). Wywołanie korzysta również z opcji GMATM, dlatego komunikaty dłuższe niż zadeklarowana wielkość buforu są obcinane.

Program demonstruje sposób, w jaki należy usunąć pola *MDMID* i *MDCOD* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie ustawia te pola na wartości zawarte w komunikacie, który

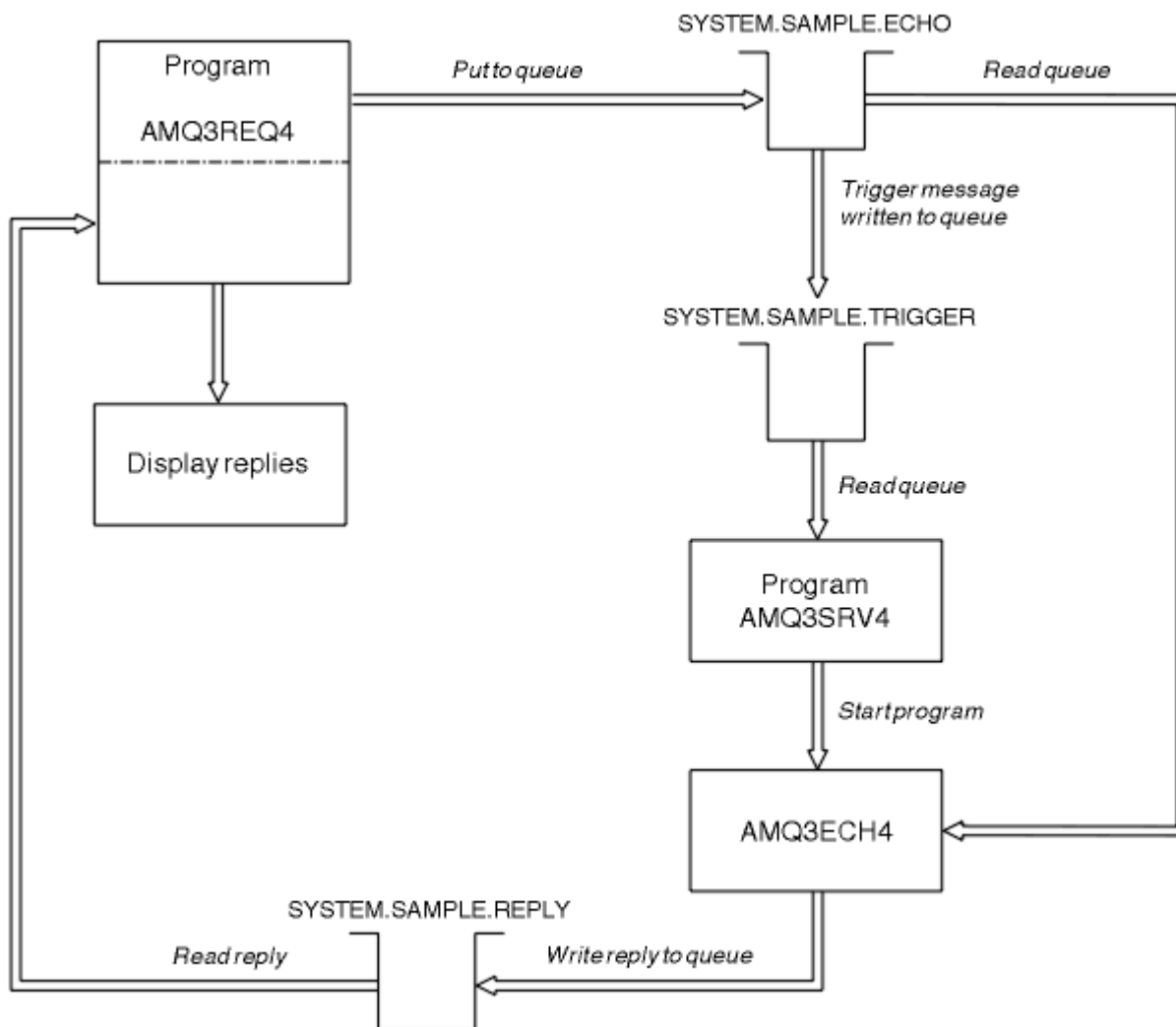
pobiera. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej komunikaty są przechowywane w kolejce.

Program będzie kontynuowany do czasu, aż wywołanie MQGET zwróci kod przyczyny RC2033 (brak dostępnego komunikatu) lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, w programie zostanie wyświetlony komunikat o błędzie zawierający kod przyczyny.

Następnie program zamyka kolejkę serwera i kolejkę zwrotną przy użyciu wywołania MQCLOSE. Tabela 180 na stronie 1449 przedstawia zmiany w przykładowym programie Echo, które są niezbędne do uruchomienia programów przykładowych Inquire i Set.

Uwaga: Szczegółowe informacje na temat przykładowego programu Echo są zawarte jako odniesienie.

Tabela 180. Przykładowy program klienta/serwera-szczegóły		
Nazwa programu	Kolejka SYSTEM/SAMPLE	Program został uruchomiony
Echo	ECHO	AMQ3ECH4
Zapytania	INQ	AMQ3INQ4
Zestaw	SET	AMQ3SET4



Rysunek 9. Wykres blokowy programu przykładowego klienta/serwera (Echo)

Korzystanie z wyzwalania za pomocą przykładu Żądanie

Aby uruchomić przykład przy użyciu wyzwalania, uruchom program serwera wyzwalacza, AMQ3SRV4, w przypadku wymaganej kolejki inicjuj w jednym zadaniu, a następnie uruchom komendę AMQ3REQ4 w innym zadaniu.

Oznacza to, że serwer wyzwalacza jest gotowy, gdy przykładowy program żądania wysyła komunikat.

Uwaga:

1. W przykładach używana jest kolejka SAMPLE TRIGGER jako kolejka inicjujący dla pliku SYSTEM.SAMPLE.ECHO, SYSTEM.SAMPLE.INQ lub SYSTEM.SAMPLE.SET kolejek lokalnych. Alternatywnie można zdefiniować własną kolejkę inicjującą.
2. Przykładowe definicje utworzone przez komendę AMQSAMP4 powodują wyzwolenie wersji C próbki. Aby wywołać wersję RPG, należy zmienić definicje procesów SYSTEM.SAMPLE.ECHOPROCESS i SYSTEM.SAMPLE.INQPROCESS i SYSTEM.SAMPLE.SETPROCESS. Aby wykonać tę operację, można użyć komendy CHGMQMPRC (więcej szczegółowych informacji na ten temat zawiera sekcja [Change MQ Process \(CHGMQMPRC\)](#)), a następnie przeprowadzić edycję i uruchomić własną wersję produktu AMQSAMP4.
3. Należy skompilować program uruchamiający serwer wyzwalany ze źródła podanego w QMQMSAMP/QRPGLESRC.

W zależności od procesu wyzwalacza, który ma zostać uruchomiony, należy wywołać komendę AMQ3REQ4 z parametrem określaniem komunikatów żądań, które mają zostać umieszczone w jednej z tych przykładowych kolejek serwera:

- SYSTEM.SAMPLE.ECHO (dla przykładowych programów Echo)
- SYSTEM.SAMPLE.INQ (dla przykładowych programów Inquire)
- SYSTEM.SAMPLE.SET (w przypadku zestawu przykładowych programów)

Wykres przepływu dla zmiennej SYSTEM.SAMPLE.ECHO jest wyświetlany w programie [Rysunek 9 na stronie 1449](#). Za pomocą komendy, która ma wydać żądanie programu RPG na ten serwer, jest:

```
CALL PGM(QMQMSAMP/AMQ3REQ4) PARM('SYSTEM.SAMPLE.ECHO
+ 30 blank characters','Queue_Manager_Name')
```

ponieważ nazwa kolejki i nazwa menedżera kolejek *muszą* mieć 48 znaków długości.

Uwaga: Ta kolejka przykładowa ma typ wyzwalacza FIRST, więc jeśli przed uruchomieniem przykładu żądania są już komunikaty w kolejce, aplikacje serwera nie są wyzwalane przez wysyłane komunikaty.

Jeśli chcesz spróbować dalszych przykładów, możesz spróbować następujących wariantów:

- Użyj komendy AMQ3TRG4 zamiast AMQ3SRV4, aby zamiast tego wprowadzić zadanie, ale ewentualne opóźnienia w przedłożeniu zadania mogą sprawić, że będzie mniej łatwe w podążaniu za tym, co się dzieje.
- Użyj komendy SYSTEM.SAMPLE.INQ i SYSTEM.SAMPLE.SET. Korzystając z przykładowego pliku danych, komendy służące do wydawania żądań programów RPG do tych serwerów są następujące:

```
CALL PGM(QMQMSAMP/AMQ3INQ4) PARM('SYSTEM.SAMPLE.INQ
+ 31 blank characters')
CALL PGM(QMQMSAMP/AMQ3SET4) PARM('SYSTEM.SAMPLE.SET
+ 31 blank characters')
```

ponieważ nazwa kolejki *musi* mieć długość 48 znaków.

Te przykładowe kolejki również mają typ wyzwalacza FIRST.

Przykładowy program Echo w systemie IBM i

Przykładowe programy Echo zwracają komunikat wysłany do kolejki odpowiedzi. Program ma nazwę AMQ3ECH4

Aby proces wyzwalający działał, należy upewnić się, że przykładowy program Echo, który ma być używany, jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.ECHO. W tym celu należy podać nazwę przykładowego programu Echo, który ma być używany w polu *AppLId* definicji procesu SYSTEM.SAMPLE.ECHOPROCESS. (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji [Administrowanie programem IBM i](#)). Kolejka przykładowa ma typ wyzwalacza FIRST, więc jeśli przed uruchomieniem przykładu żądania w kolejce znajdują się już komunikaty, to próba Echo nie jest wyzwalana przez wysyłane komunikaty.

Jeśli definicja została ustawiona poprawnie, należy najpierw uruchomić komendę AMQ3SRV4 w jednym zadaniu, a następnie uruchomić komendę AMQ3REQ4 w innym. Można użyć komendy AMQ3TRG4 zamiast AMQ3SRV4, ale potencjalne opóźnienia w składaniu zadań mogą spowodować, że śledzenie zdarzeń będzie mniej proste.

Aby wysłać komunikaty do kolejki SYSTEM.SAMPLE.ECHO. Przykładowe programy Echo wysyłają komunikat odpowiedzi zawierający dane zawarte w komunikacie żądania do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Echo

Po wyzwoleniu programu jawnie łączy się on z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Mimo że nie jest to konieczne w przypadku produktu IBM MQ for IBM i, oznacza to, że można użyć tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. W tym wywołaniu używane są opcje GMATM i GMWT, z interwałem oczekiwania 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program korzysta z wywołania MQPUT w celu umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera treść komunikatu żądania.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłączy się z menedżerem kolejek.

Ten program może również odpowiadać na komunikaty wysłane do kolejki z platform innych niż IBM MQ for IBM i, chociaż dla tej sytuacji nie jest dostarczana żadna próba. Aby program ECHO działał, należy wykonać następujące czynności:

- Napisz program, poprawnie określając pola *Format*, *Encoding* *CCSID* , aby wysłać komunikaty żądania tekstowego.

Program ECHO żąda, aby menedżer kolejek wykonał konwersję danych komunikatów, jeśli jest to potrzebne.

- Podaj CONVERT (*YES) w kanale wysyłającym IBM MQ for IBM i , jeśli napisany przez Ciebie program nie zapewnia podobnej konwersji dla odpowiedzi.

Przykładowy program Inquire w systemie IBM i

Przykładowy program Inquire, AMQ3INQ4, zawiera informacje na temat niektórych atrybutów kolejki przy użyciu wywołania MQINQ.

Program jest przeznaczony do uruchamiania jako program wyzwalany, więc jego jedynym wejściem jest struktura MQTMC (komunikat wyzwalacza). Ta struktura zawiera nazwę kolejki docelowej z atrybutami, które mają zostać zapytane.

Aby proces wyzwalający działał, należy upewnić się, że przykładowy program Inquire jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.INQ. Aby wykonać to, należy podać nazwę przykładowego programu Inquire w polu *AppLId* w pliku SYSTEM.SAMPLE.INQPROCESS , definicja procesu. (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji [Change MQ Process](#))

(CHGMQMPRC)). Kolejka przykładowa ma typ wyzwalacza FIRST, więc jeśli przed uruchomieniem przykładu żądania w kolejce znajdują się już komunikaty, to próba zapytania nie jest wyzwalana przez wysyłane komunikaty.

Jeśli definicja została ustawiona poprawnie, należy najpierw uruchomić komendę AMQ3SRV4 w jednym zadaniu, a następnie uruchomić komendę AMQ3REQ4 w innym. Można użyć komendy AMQ3TRG4 zamiast AMQ3SRV4, ale potencjalne opóźnienia w składaniu zadań mogą spowodować, że śledzenie zdarzeń będzie mniej proste.

Za pomocą przykładowego programu żądania można wysyłać komunikaty żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.INQ. Dla każdego komunikatu żądania program przykładowy Inquire wysyła komunikat odpowiedzi zawierający informacje na temat kolejki określonej w komunikacie żądania. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Inquire

Po wyzwoleniu programu jawnie łączy się on z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Chociaż nie jest to konieczne dla produktu IBM MQ for IBM i, ta funkcja projektowania oznacza, że można używać tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. W tym wywołaniu używane są opcje GMATM i GMWT, z interwałem oczekiwania 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (którą wywołamy *kolejkę docelową*). zawarte w danych i otwierają tę kolejkę za pomocą wywołania MQOPEN z opcją OOINQ. Następnie program korzysta z wywołania MQINQ w celu sprawdzenia wartości atrybutów *InhibitGet*, *CurrentQDepth* i *OpenInputCount* w kolejce docelowej.

Jeśli wywołanie MQINQ powiedzie się, program użyje wywołania MQPUT w celu umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera wartości trzech atrybutów.

Jeśli wywołanie MQOPEN lub MQINQ nie powiodło się, program korzysta z wywołania MQPUT w celu umieszczenia komunikatu *report* w kolejce odpowiedzi. W polu *MDFB* deskryptora komunikatu tego komunikatu raportu jest to kod przyczyny zwracany przez wywołanie MQOPEN lub MQINQ, w zależności od tego, który błąd nie powiódł się.

Po wywołaniu wywołania MQINQ program zamknie kolejkę docelową przy użyciu wywołania MQCLOSE.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłączy się z menedżerem kolejek.

Przykładowy program Set w systemie IBM i

Program przykładowy Set, AMQ3SET4, hamuje operacje umieszczania w kolejce przy użyciu wywołania MQSET w celu zmiany atrybutu *InhibitPut* kolejki.

Program jest przeznaczony do uruchamiania jako program wyzwalany, więc jego jedynym wejściem jest struktura MQTMC (komunikat wyzwalacza), która zawiera nazwę kolejki docelowej z atrybutami, które mają zostać zapytane.

Aby proces wyzwalający działał, należy upewnić się, że przykładowy program Set został wyzwolony przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.SET. W tym celu należy podać nazwę przykładowego programu Set w polu *AppId* definicji procesu SYSTEM.SAMPLE.SETPROCESS. (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji [Administrowanie programem IBM i](#)). Kolejka przykładowa ma typ wyzwalacza FIRST, więc jeśli przed uruchomieniem przykładu żądania w kolejce znajdują się już komunikaty, to próbka zestawu nie jest wyzwalana przez wysyłane komunikaty.

Jeśli definicja została ustawiona poprawnie, należy najpierw uruchomić komendę AMQ3SRV4 w jednym zadaniu, a następnie uruchomić komendę AMQ3REQ4 w innym. Można użyć komendy AMQ3TRG4 zamiast AMQ3SRV4, ale potencjalne opóźnienia w składaniu zadań mogą spowodować, że śledzenie zdarzeń będzie mniej proste.

Za pomocą przykładowego programu żądania można wysyłać komunikaty żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.SET. Dla każdego komunikatu żądania, program przykładowy Ustaw wysyła komunikat odpowiedzi zawierający potwierdzenie, że operacje put zostały zablokowane w określonej kolejce. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Set

Po wyzwoleniu programu jawnie łączy się on z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Mimo że nie jest to konieczne w przypadku produktu IBM MQ for IBM i, oznacza to, że można użyć tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę nazwaną w strukturze komunikatu wyzwalacza, która została przekazana podczas jej uruchamiania. (Dla jasności zadzwonimy do tej *kolejki żądań*.) Program korzysta z wywołania MQOPEN, aby otworzyć tę kolejkę dla współużytkowanych danych wejściowych.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. W tym wywołaniu używane są opcje GMATM i GMWT, z interwałem oczekiwania 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania. Jeśli tak nie jest, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (którą wywołamy *kolejkę docelową*), zawarte w danych i otwierają tę kolejkę za pomocą wywołania MQOPEN z opcją OOSSET. Następnie program korzysta z wywołania MQSET w celu ustawienia wartości atrybutu *InhibitPut* kolejki docelowej na wartość QAPUTI.

Jeśli wywołanie MQSET zakończy się pomyślnie, program korzysta z wywołania MQPUT w celu umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera łańcuch PUT inhibited.

Jeśli wywołanie MQOPEN lub MQSET nie powiodło się, program korzysta z wywołania MQPUT w celu umieszczenia komunikatu *raport* w kolejce odpowiedzi. W polu *MDFB* deskryptora komunikatu tego komunikatu raportu jest to kod przyczyny zwracany przez wywołanie MQOPEN lub MQSET, w zależności od tego, który błąd nie powiódł się.

Po wywołaniu komendy MQSET program zamknie kolejkę docelową przy użyciu wywołania MQCLOSE.

Jeśli w kolejce żądań nie pozostały żadne komunikaty, program zamknie tę kolejkę i rozłącza się z menedżerem kolejek.

Przykładowe programy wyzwalające w systemie IBM i

Produkt IBM MQ for IBM i udostępnia dwa przykładowe programy wyzwalające, które są zapisywane w języku ILE/RPG.

Programy są następujące:

AMQ3TRG4

Jest to monitor wyzwalacza dla środowiska IBM i. Wprowadza ona zadanie IBM i do uruchomienia aplikacji, ale oznacza to, że istnieje dodatkowy koszt przetwarzania powiązany z każdym komunikatem wyzwalacza.

AMQ3SRV4

Jest to serwer wyzwalacza dla środowiska IBM i. Dla każdego komunikatu wyzwalacza serwer uruchamia komendę uruchamiania w swoim własnym zadaniu, aby uruchomić określoną aplikację. Serwer wyzwalacza może wywoływać transakcje CICS.

Wersje językowe tych przykładów są również dostępne jako programy wykonywalne w bibliotece QMQM o nazwach AMQSTRG4 i AMQSERV4.

Przykładowy monitor wyzwalacza AMQ3TRG4 w systemie IBM i

AMQ3TRG4 to monitor wyzwalacza. Przyjmuje on jeden parametr: nazwę kolejki inicjuj, która ma służyć. AMQSAMP4 definiuje przykładową kolejkę inicjującą, SYSTEM.SAMPLE.TRIGGER, którego można użyć przy próbie próby programów przykładowych.

Komenda AMQ3TRG4 wprowadza zadanie IBM i dla każdego poprawnego komunikatu wyzwalacza, który pobiera z kolejki inicjuj.

Projekt monitora wyzwalacza

Monitor wyzwalacza otwiera kolejkę inicjującą i pobiera komunikaty z kolejki, określając nieograniczony przedział czasu oczekiwania.

Monitor wyzwalacza wprowadza zadanie IBM i w celu uruchomienia aplikacji określonej w komunikacie wyzwalacza i przekazuje strukturę MQTMC (wersja znakowa komunikatu wyzwalacza). Dane środowiska zawarte w komunikacie wyzwalacza są używane jako parametry wprowadzania zadania.

Na koniec program zamyka kolejkę inicjującą.

Przykładowy serwer wyzwalacza AMQ3SRV4

AMQ3SRV4 to serwer wyzwalacza. Przyjmuje on jeden parametr: nazwę kolejki inicjuj, która ma służyć. AMQSAMP4 definiuje przykładową kolejkę inicjującą, SYSTEM.SAMPLE.TRIGGER, którego można użyć przy próbie próby programów przykładowych.

Dla każdego komunikatu wyzwalacza AMQ3SRV4 uruchamia komendę uruchomienia w swoim własnym zadaniu w celu uruchomienia określonej aplikacji.

Za pomocą przykładowej kolejki wyzwalacza, której komenda ma zostać wydana, jest następująca komenda:

```
CALL PGM(QMQM/AMQ3SRV4) PARM('Queue Name')
```

Gdzie Queue Name *musi* mieć 48 znaków długości, co jest osiąganym przez dopełnienie nazwy kolejki z wymaganą liczbą odstępów. Dlatego też, jeśli używany jest plik SYSTEM.SAMPLE.TRIGGER jako kolejka docelowa, należy mieć 28 pustych znaków.

Projekt serwera wyzwalacza

Projekt serwera wyzwalającego jest podobny do projektu monitora wyzwalacza, z wyjątkiem serwera wyzwalaczy:

- Umożliwia CICS oraz aplikacjom produktu IBM i
- Nie używa danych środowiska z komunikatu wyzwalacza.
- Wywołuje aplikacje IBM i we własnym zadaniu (lub używa funkcji STRCICSUSR do uruchamiania aplikacji CICS), zamiast wprowadzania zadania IBM i.
- Otwiera kolejkę inicjującą dla współużytkowanych danych wejściowych, tak więc wiele serwerów wyzwalanych może być uruchomione w tym samym czasie

Uwaga: Programy uruchomione przez program AMQ3SRV4 nie mogą korzystać z wywołania MQDISC, ponieważ spowoduje to zatrzymanie serwera wyzwalacza. Jeśli programy uruchomione przez program AMQ3SRV4 korzystają z wywołania MQCONN, uzyskają kod przyczyny RC2002.

Kończenie przykładowych programów wyzwalających w systemie IBM i

Program monitora wyzwalacza może zostać zakończony przez opcję 2 (ENDRQS) sysrequest lub przez zahamowanie pobierania z kolejki wyzwalacza.

Jeśli przykładowa kolejka wyzwalaczy jest używana, komenda jest następująca:

```
CHGMQM QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*NO)
```

Uwaga: Aby ponownie uruchomić wyzwalanie w tej kolejce, należy użyć komendy *musi* wprowadzić komendę:

```
CHGMQM QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Uruchamianie przykładów przy użyciu kolejek zdalnych w systemie IBM i

Istnieje możliwość zademonstrowania zdalnego kolejkowania przez uruchomienie przykładów w połączonych menedżerach kolejek komunikatów.

Program AMQSAMP4 udostępnia lokalną definicję kolejki zdalnej (SYSTEM.SAMPLE.REMOTE), w którym używany jest zdalny menedżer kolejek o nazwie OTHER. Aby użyć tej definicji przykładu, należy zmienić OTHER na nazwę drugiego menedżera kolejek komunikatów, który ma być używany. Należy również skonfigurować kanał komunikatów między dwoma menedżerami kolejek komunikatów. Aby uzyskać informacje na temat tego, jak to zrobić, należy zapoznać się z informacjami na temat [programów obsługi wyjścia kanału dla kanałów przesyłania komunikatów](#).

Przykładowy program żądania umieszcza własną nazwę lokalnego menedżera kolejek w polu *MDRM* wysyłanych przez niego komunikatów. Przykłady Inquire i Set wysyłają komunikaty odpowiedzi do kolejki i menedżera kolejek komunikatów o nazwie w polach *MDRQ* i *MDRM* w komunikatach żądań, które przetwarzali.

Kody powrotu dla IBM i (ILE RPG)

Te informacje opisują kody powrotu powiązane z interfejsem MQI i MQAI.

Kody powrotu powiązane z:

- Komendy PCF (Programmable Command Format) są wymienione w sekcji [Skorowidz formatów komend programowalnych](#).
- Wywołania C + + są wymienione w [Używanie języka C++](#).

W przypadku każdego wywołania kod zakończenia i kod przyczyny są zwracane przez menedżer kolejek lub przez procedurę wyjścia w celu wskazania powodzenia lub niepowodzenia wywołania.

Aplikacje nie mogą zależeć od błędów, które są sprawdzane w określonej kolejności, z wyjątkiem przypadków, w których zaznaczono inaczej. Jeśli z wywołania może powstać więcej niż jeden kod zakończenia lub kod przyczyny, konkretny zgłoszony błąd zależy od implementacji.

Kody zakończenia dla IBM i (ILE RPG)

Parametr kodu zakończenia (*CMPCOD*) pozwala programowi wywołującemu na szybkie sprawdzenie, czy wywołanie zostało zakończone pomyślnie, zakończone częściowo lub nie powiodło się.

CCOK

(MQCC_OK na innych platformach)

Zakończenie powiodło się.

Wywołanie zakończyło się całkowicie; wszystkie parametry wyjściowe zostały ustawione. W tym przypadku parametr *REASON* zawsze ma wartość RCNONE.

CCWARN

(MQCC_WARN na innych platformach)

Ostrzeżenie (częściowe zakończenie).

Połączenie zostało zakończone częściowo. Niektóre parametry wyjściowe mogły zostać ustawione jako uzupełnienie parametrów wyjściowych *CMPCOD* i *REASON*. Parametr *REASON* zawiera dodatkowe informacje o częściowym zakończeniu.

CCFAIL

(MQCC_FAIL na innych platformach)

Wywołanie nie powiodło się.

Przetwarzanie wywołania nie zostało zakończone, a stan menedżera kolejek jest normalnie niezmieniony. Wyjątki są szczególnie zauważalne. Parametry wyjściowe *CMPCOD* i *REASON* zostały ustawione; pozostałe parametry są niezmienione, z wyjątkiem sytuacji, w których zaznaczono.

Przyczyną może być błąd w programie użytkowym, lub może to być rezultat niektórych sytuacji zewnętrznych względem programu, na przykład uprawnienia użytkownika mogły zostać odwołane. Parametr *REASON* zawiera dodatkowe informacje na temat błędu.

Kody przyczyny dla IBM i (ILE RPG)

Parametr kodu przyczyny (*REASON*) jest kwalifikacją do parametru kodu zakończenia (*CMPCOD*).

Jeśli nie ma specjalnego powodu do raportowania, zwracana jest wartość RCNONE. Pomyślne wywołanie zwraca CCOK i RCNONE.

Jeśli kodem zakończenia jest CCWARN lub CCFAIL, menedżer kolejek zawsze zgłasza odpowiedni powód; szczegóły są podawane w każdym opisie wywołania.

W przypadku, gdy procedury obsługi wyjścia użytkownika ustawiają kody zakończenia i przyczyny, powinny stosować się do tych reguł. Ponadto wszystkie specjalne wartości przyczyny zdefiniowane przez procedury zewnętrzne powinny być mniejsze od zera, aby zapewnić, że nie będą one kolidowały z wartościami zdefiniowanymi przez menedżer kolejek. Wyjścia mogą ustawiać przyczyny już zdefiniowane przez menedżer kolejek, o ile są one odpowiednie.

Kody przyczyny występują również w:

- Pole *DLREA* struktury MQDLH
- Pole *MDFB* struktury MQMD

Pełna lista kodów przyczyn znajduje się w sekcji [Kody zakończenia i przyczyny API](#).

Aby znaleźć kod przyczyny produktu IBM i na tej liście, należy usunąć "RC" z przodu, na przykład RC2002 staje się 2002. Ponadto kody zakończenia są wyświetlane w postaci, w której znajdują się na innych platformach:

Tabela 181.	
IBM i	Inne platformy
CCOK	MQCC_OK
CCWARN	MQCC_WARN
CCFAIL	MQCC_FAIL

Reguły sprawdzania poprawności opcji MQI dla produktu IBM i (ILE RPG)

Ten temat zawiera informacje na temat sytuacji, które generują kod przyczyny RC2046 z wywołania MQOPEN, MQPUT, MQPUT1, MQGET lub MQCLOSE.

Wywołanie MQOPEN w systemie IBM i

Dla opcji wywołania MQOPEN:

- *Należy określić co najmniej jeden z następujących elementów:*
 - OOBROW
 - POINPQ
 - OOINPX
 - OOINPS
 - OOINQ

- OOOOUT
- OOSSET
- Dozwolony jest tylko *jeden* z następujących:
 - POINPQ
 - OOINPX
 - OOINPS
- Dozwolony jest tylko *jeden* z następujących:
 - OOBNDQ
 - OOBNDN
 - OOBNDQ

Uwaga: Opcje wymienione wcześniej wykluczają się wzajemnie. Ponieważ jednak wartość OOBNDQ wynosi zero, określenie jej przy użyciu jednej z dwóch pozostałych opcji wiązania nie powoduje kodu przyczyny RC2046. OOBNDQ jest dostarczane do dokumentacji programu pomocy.

- Jeśli określono parametr OOSAVA, należy również określić jedną z opcji OOINP*.
- Jeśli zostanie podany jeden z opcji OOSSET* lub OOPAS*, należy również określić wartość OOOOUT.

Wywołanie MQPUT w systemie IBM i

Dla opcji put-message:

- Kombinacja PMSYP i PMNSYP nie jest dozwolona.
- Dozwolony jest tylko *jeden* z następujących:
 - PMDEFC
 - PMNOC
 - PMPASA
 - PMPASI
 - PMSETA
 - PMSETI
- Parametr PMALTU nie jest dozwolony (jest poprawny tylko w wywołaniu MQPUT1).

Wywołanie MQPUT1 w systemie IBM i

W przypadku opcji put-message reguły są takie same, jak dla wywołania MQPUT, z wyjątkiem następujących opcji:

- PMALTU jest dozwolone.
- PMLOGO *nie* jest dozwolone.

Wywołanie MQGET w systemie IBM i

Dla opcji get-message:

- Dozwolona jest tylko *jedna* z następujących opcji:
 - GMNSYP
 - GMSYP
 - GMPSYP
- Dozwolona jest tylko *jedna* z następujących opcji:
 - GMBRWF

- GMBRWC
- GMBRWN
- GMMUC
- Program GMSYP nie jest dozwolony z żadną z następujących opcji:
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMLK
 - GMUNLK
- Program GMPSTYP nie jest dozwolony z żadną z następujących opcji:
 - GMBRWF
 - GMBRWC
 - GMBRWN
 - GMCMPM
 - GMUNLK
- Jeśli określono parametr GMLK, należy również określić jedną z następujących opcji:
 - GMBRWF
 - GMBRWC
 - GMBRWN
- Jeśli określono parametr GMUNLK, dozwolone są tylko następujące opcje:
 - GMNSTYP
 - GMNWT

Wywołanie MQCLOSE w systemie IBM i

- W przypadku opcji wywołania MQCLOSE. Kombinacja kodów CODEL i COPURG nie jest dozwolona.
- Dozwolona jest tylko jedna z następujących wartości:
 - COKPSB
 - CORMSB

Wywołanie MQSUB w systemie IBM i

W przypadku opcji wywołania MQSUB:

- Należy określić co najmniej jedną z następujących wartości:
- Należy określić co najmniej jedną z następujących wartości:
 - SOALT
 - SORES
 - SOCRT
- Dozwolona jest tylko jedna z następujących wartości:
 - SODUR
 - SONDUR

Uwaga: Opcje wymienione wcześniej wykluczają się wzajemnie. Jednak ponieważ wartość parametru SONDUR wynosi zero, określenie jej przy użyciu parametru SODUR nie powoduje kodu przyczyny RC2046. SONDUR jest dostarczany do dokumentacji programu pomocowego.

- Kombinacja SOGRP i SOMAN nie jest dozwolona.

- SOGRP wymaga określenia SOSCID.
- Dozwolony jest tylko jeden z następujących elementów: SOAUID SOFUID
- Kombinacja SONEWP i SOPUBR nie jest dozwolona.
- SONEWP jest dozwolone tylko w połączeniu z SOCRT.
- Dozwolona jest tylko jedna z następujących wartości:
 - SOWCHR
 - SOWTOP

Kodowanie komputera w systemie IBM i

Te informacje umożliwiają zapoznanie się z strukturą pola *MDENC* w deskrytorze komunikatu.

Więcej informacji na temat deskryptora komunikatu zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124.

Pole *MDENC* jest 32-bitową liczbą całkowitą, która jest podzielona na cztery oddzielne podpola. Te podpola identyfikują:

- Kodowanie używane dla binarnych liczb całkowitych
- Kodowanie używane dla upakowanych liczb całkowitych
- Kodowanie używane dla liczb zmiennopozycyjnych
- Zarezerwowane bity

Każde podpole jest identyfikowane za pomocą maski bitowej, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Zdefiniowane są następujące maski:

PLIMSK

Maska dla kodowania binarnego-liczba całkowita.

To podpole zajmuje pozycje od 28 do 31 w obrębie pola *MDENC*.

ENDMSK

Maska dla kodowania spakowanego-dziesiętnego.

To podpole zajmuje pozycje od 24 do 27 w obrębie pola *MDENC*.

ENFMSK

Maska dla kodowania zmiennopozycyjnego.

To podpole zajmuje pozycje od 20 do 23 w obrębie pola *MDENC*.

ENRMSK

Maska dla bitów zarezerwowanych.

To podpole zajmuje pozycje od 0 do 19 w obrębie pola *MDENC*.

Kodowanie binarno-całkowite

Poprawne wartości dla kodowania binarnego-liczba całkowita.

Następujące wartości są poprawne dla kodowania binarnego-liczba całkowita:

OKR.

Niezdefiniowane kodowanie całkowite.

Binarne liczby całkowite są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

ENINOR

Normalne kodowanie liczb całkowitych.

Binarne liczby całkowite są reprezentowane w tradycyjny sposób:

- Najmniej znaczący bajt w tym numerze ma najwyższy adres dowolnego z bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie znajduje się obok bajtu o następnym wyższym adresie; najbardziej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym dolnym adresem.

ENIREV

Odwrócone kodowanie całkowite.

Binarne liczby całkowite są reprezentowane w taki sam sposób, jak ENINOR, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są rozmieszczone w taki sam sposób jak ENINOR.

Packed-decimal-kodowanie całkowite

Poprawne wartości dla kodowania w upakowanym formacie dziesiętnym

Następujące wartości są poprawne dla kodowania spakowanego-dziesiętnego-integer:

OKRAQ.

Niezdefiniowane kodowanie upakowane dziesiętne.

Upakowane dziesiętne liczby całkowite są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

ENDNOR

Normalne kodowanie po przecinku.

Spakowane-dziesiętne liczby całkowite są reprezentowane w tradycyjny sposób:

- Każda cyfra dziesiętna w postaci drukowalnej liczby jest reprezentowana w upakowanym formacie dziesiętnym przez pojedynczą cyfrę szesnastkową z zakresu od X' 0 'do X' 9'. Każda cyfra szesnastkowa zajmuje 4 bity, a więc każdy bajt w upakowanej liczbie dziesiętnej reprezentuje dwie cyfry dziesiętne w postaci drukowalnej liczby.
- Najmniej znaczącym bajtem w upakowanej liczbie dziesiętnej jest bajt, który zawiera najmniej znaczącą cyfrę dziesiętną. W tym bajcie, najbardziej znaczące 4 bity zawierają najmniej znaczącą cyfrę dziesiętną, a najmniej znaczące 4 bity zawierają znak. Znakiem jest X'C '(dodatni), X 'D' (ujemny) lub X'F ' (niepodpisany).
- Najmniej znaczący bajt w tym numerze ma najwyższy adres dowolnego z bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie znajduje się obok bajtu o następnym wyższym adresie; najbardziej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym dolnym adresem.

ENDREV

Odwrócone kodowanie po przecinku dziesiętnym.

Spakowane liczby całkowite są reprezentowane w ten sam sposób co ENDNOR, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są układane w taki sam sposób jak ENDNOR.

Kodowanie zmiennopozycyjne

Poprawne wartości dla kodowania zmiennopozycyjnego

Następujące wartości są poprawne dla kodowania zmiennopozycyjnego:

ENFUND

Niezdefiniowane kodowanie zmiennopozycyjne.

Liczby zmiennopozycyjne są reprezentowane przy użyciu kodowania, które jest niezdefiniowane.

ENFNOR

Normal IEEE (The Institute of Electrical and Electronics Engineers) float encoding.

Liczby zmiennoprzecinkowe są reprezentowane przy użyciu standardowego formatu zmiennopozycyjnego IEEE z liczbą bajtów ustawionych w następujący sposób:

- Najmniej znaczący bajt w mantysie ma najwyższy adres dowolnego z bajtów w liczbie; bajt zawierający wykładnik ma najniższy adres
- Najmniej znaczący bit w każdym bajcie znajduje się obok bajtu o kolejnym wyższym adresie; najbardziej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym dolnym adresem

Szczegółowe informacje na temat kodowania zmiennopozycyjnego IEEE można znaleźć w standardzie IEEE 754.

ENFREV

Odwrócone kodowanie zmiennopozycyjne IEEE.

Liczby zmiennopozycyjne są reprezentowane w taki sam sposób, jak ENFNOR, ale z bajtami ułożonych w odwrotnej kolejności. Bity w każdym bajcie są układane w taki sam sposób jak ENFNOR.

ENF390

Kodowanie zmiennopozycyjne architektury System/390 .

Liczby zmiennopozycyjne są reprezentowane przy użyciu standardowego formatu zmiennopozycyjnego System/390 . Jest on używany również przez system System/370.

Konstruowanie kodowania

Aby skonstruować wartość dla pola *MDENC* w strukturze MQMD, należy dodać odpowiednie stałe opisujące wymagane kodowania.

Należy pamiętać, aby połączyć tylko jeden z kodowań ENI* z jednym z kodowań END* i jednym z kodowań ENF*.

Analizowanie kodowania

Pole *MDENC* zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić liczbę całkowitą, upakowaną liczbę dziesiętną lub kodowanie zmiennopozycyjne, powinny korzystać z techniki opisanej w tym temacie.

Korzystanie z arytmetyki

Następujące kroki powinny być wykonywane przy użyciu arytmetyki liczb całkowitych:

1. Należy wybrać jedną z następujących wartości, zgodnie z wymaganym typem kodowania:
 - 1 dla binarnego kodowania liczb całkowitych
 - 16 dla upakowanego dziesiętnego kodowania liczb całkowitych
 - 256 dla kodowania zmiennopozycyjnego

Wywołaj wartość A.

2. Podziel wartość pola *MDENC* przez A ; Wywołaj wynik B.
3. Podziel B przez 16; wywołaj wynik C.
4. Pomnóż C przez 16 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. E jest wymaganym kodowaniem i może zostać przetestowany pod kątem równości z każdą z wartości, które są poprawne dla danego typu kodowania.

Podsumowanie kodowań architektury maszyn

Tabela podsumowująca kodowania dla architektur maszyn.

Kodowania dla architektur maszyn są wyświetlane w programie [Tabela 182 na stronie 1462](#).

Tabela 182. Podsumowanie kodowań dla architektur maszyn

Architektura maszyny	Kodowanie binarnego liczb całkowitych	Pakowane-dziesiętne kodowanie całkowite	Kodowanie zmiennopozycyjne
IBM i	normalny	normalny	Normalne IEEE
Intel x86	Odwrotne	Odwrotne	IEEE odwrócone
PowerPC	normalny	normalny	Normalne IEEE
System/390	normalny	normalny	System/390

Opcje raportów i flagi komunikatów

Ten temat dotyczy pól *MDREP* i *MDMFL*, które są częścią deskryptora komunikatu *MQMD* określonego w wywołaniach *MQGET*, *MQPUT* i *MQPUT1*.

Więcej informacji na temat deskryptora komunikatu zawiera sekcja [“MQMD-deskryptor komunikatu”](#) na stronie 1124. Informacje te opisują:

- Struktura pola raportu i sposób jego przetwarzania przez menedżer kolejek
- W jaki sposób aplikacja powinna analizować pole raportu
- Struktura pola komunikatu-flagi

Struktura pola raportu

Pole *MDREP* jest 32-bitową liczbą całkowitą, która jest podzielona na trzy oddzielne podpola.

Te podpola identyfikują:

- Opcje raportu, które są odrzucane, jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Opcje raportów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Opcje raportu, które są akceptowane tylko wtedy, gdy spełnione są określone inne warunki

Każde podpole jest identyfikowane za pomocą maski bitowej, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Należy zauważyć, że bity w podpolu niekoniecznie są sąsiadujące. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Następujące maski są zdefiniowane w celu identyfikacji podpól:

RORUM

Maska dla nieobsługiwanych opcji raportu, które zostały odrzucone.

Ta maska identyfikuje pozycje bitowe w polu *MDREP*, gdzie opcje raportów, które nie są obsługiwane przez lokalny menedżer kolejek, spowodują wywołanie *MQPUT* lub *MQPUT1* w przypadku niepowodzenia kodu zakończenia *CCFAIL* i kodu przyczyny *RC2061*.

To podpole zajmuje pozycje bitowe 3, a 11 do 13.

ROAUM

Maska dla nieobsługiwanych opcji raportu, które są akceptowane.

Ta maska identyfikuje pozycje bitowe w polu *MDREP*, w których opcje raportów, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach *MQPUT* lub *MQPUT1*. W tym przypadku zwracany jest kod zakończenia *CCWARN* z kodem przyczyny *RC2104*.

To podpole zajmuje pozycje bity od 0 do 2, od 4 do 10 i od 24 do 31.

W tym podpolu znajdują się następujące opcje raportu:

- ROCMTC
- RODLQ
- RODISC
- ROEXC

- ROEXCD
- ROEXCF
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- RONONE
- ROPAN
- ROPCI
- ROPMI

ROAUXM

Maska dla nieobsługiwanych opcji raportu, które są akceptowane tylko w określonych okolicznościach.

Ta maska identyfikuje pozycje bitowe w polu *MDREP*, w których opcje raportów, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem*, że spełnione są oba poniższe warunki:

- Komunikat jest przeznaczony dla menedżera kolejek zdalnych.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (oznacza to, że kolejka identyfikowana przez pola *ODMN* i *ODON* w deskrytorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest kolejką lokalną transmisji).

Kod zakończenia CCWARN z kodem przyczyny RC2104 jest zwracany, jeśli te warunki są spełnione, a wartość CCFAIL z kodem przyczyny RC2061, jeśli nie jest spełniony.

To podpole zajmuje pozycje bity od 14 do 23.

W tym podpolu znajdują się następujące opcje raportu:

- ROCOA
- ROCOAD
- ROCOAF
- ROCOD
- ROCODD
- ROCODF

Jeśli w polu *MDREP* są określone opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza poszczególne podpole z kolei za pomocą operacji bitowych AND, aby połączyć pole *MDREP* z maską dla tego podpole. Jeśli wynik tej operacji nie jest zerowy, zwracane są kody zakończenia i kody przyczyny opisane wcześniej.

Jeśli zwracana jest wartość CCWARN, nie jest ona zdefiniowana, który kod przyczyny jest zwracany, jeśli istnieją inne warunki ostrzeżenia.

Możliwość określenia i zaakceptowaniu opcji raportu, które nie są rozpoznawane przez lokalny menedżer kolejek, jest użyteczna, gdy konieczne jest wysłanie komunikatu z opcją raportu, która zostanie rozpoznana i przetworzona przez *zdalny* menedżer kolejek.

Analizowanie pola raportu

Pole MDREP zawiera podpole. W związku z tym niektóre aplikacje muszą sprawdzić, czy nadawca komunikatu zażądał określonego raportu. Aplikacje te powinny korzystać z techniki opisanej w tym temacie.

Korzystanie z arytmetyki

Następujące kroki powinny być wykonywane przy użyciu arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, zgodnie z typem raportu, który ma zostać sprawdzony:

- Raport ROCOA dla COA
- Raport ROCOD dla COD
- Raport ROEXC dla raportu o wyjątkach
- ROEXP-raport o utracie ważności

Wywołaj wartość A.

2. Divide the *MDREP* field by A ; call the result B.

3. Podziel B przez 8 ; Wywołaj wynik C.

4. Pomnóż C przez 8 i odejmij od B ; Wywołaj wynik D.

5. Pomnóż D przez A ; Wywołaj wynik E.

6. Przetestuj E , aby uzyskać równość z każdą z wartości, które są możliwe dla tego typu raportu.

Na przykład, jeśli A to ROEXC, testuj E na równość z każdym z poniższych, aby określić, co zostało określone przez nadawcę wiadomości:

- RONONE
- ROEXC
- ROEXCD
- ROEXCF

Testy mogą być wykonywane w dowolnej kolejności, co jest najbardziej wygodne dla logiki aplikacji.

Poniższy pseudocode ilustruje tę technikę dla komunikatów o wyjątkach:

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Podobną metodę można użyć do przetestowania dla opcji ROPMI lub ROPCI; należy wybrać wartość A , w zależności od tego, która z tych dwóch stałych jest odpowiednia, a następnie postępować zgodnie z opisem poprzednio, ale zastępując wartość 8 w poprzednich krokach wartością 2.

Struktura pola komunikatu-flagi

Pole *MDMFL* jest 32-bitową liczbą całkowitą, która jest podzielona na trzy oddzielne podpola.

Te podpola identyfikują:

- Flagi komunikatów, które są odrzucane, jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Flagi komunikatów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek nie rozpoznaje ich
- Flagi komunikatów, które są akceptowane tylko wtedy, gdy spełnione są pewne inne warunki

Uwaga: Wszystkie podpola w programie *MDMFL* są zarezerwowane do użycia przez menedżer kolejek.

Każde podpole jest identyfikowane za pomocą maski bitowej, która ma 1-bity w pozycjach odpowiadających podpolu, a 0-bity gdzie indziej. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 jest najmniej znaczący. Następujące maski są zdefiniowane w celu identyfikacji podpól:

MFRUM

Maska dla nieobsługiwanych flag komunikatu, które zostały odrzucone.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , gdzie flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, spowodują, że wywołanie MQPUT lub MQPUT1 zakończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2249.

To podpole zajmuje pozycje bity od 20 do 31.

W tym podpolu znajdują się następujące opcje komunikatów:

- MFLMIG
- MFLSEG
- MFMIG
- MFSEG
- MFSEGA
- MFSEGI

MFAUM

Maska dla nieobsługiwanych flag komunikatów, które są akceptowane.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach MQPUT lub MQPUT1 . Kodem zakończenia jest CCOK.

To podpole zajmuje pozycje bitowe od 0 do 11.

MFAUXM

Maska dla nieobsługiwanych flag komunikatów, które są akceptowane tylko w określonych okolicznościach.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem* , że spełnione są oba poniższe warunki:

- Komunikat jest przeznaczony dla menedżera kolejek zdalnych.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (oznacza to, że kolejka identyfikowana przez pola *ODMN* i *ODON* w deskrypcji obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest kolejką lokalną transmisji).

Kod zakończenia CCOK jest zwracany, jeśli warunki te są spełnione, a CCFAIL z kodem przyczyny RC2249 , jeśli nie.

To podpole zajmuje pozycje bitowe od 12 do 19.

Jeśli w polu *MDMFL* są określone opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza poszczególne podpola z kolei za pomocą operacji bitowych AND, aby połączyć pole *MDMFL* z maską dla tego podpola. Jeśli wynik tej operacji nie jest zerowy, zwracane są kody zakończenia i kody przyczyny opisane wcześniej.

Konwersja danych

W tym temacie opisano interfejs do wyjścia konwersji danych oraz przetwarzanie wykonywane przez menedżer kolejek, gdy wymagana jest konwersja danych.

Wyjście konwersji danych jest wywoływane jako część procesu przetwarzania wywołania MQGET w celu przekształcenia danych komunikatu aplikacji w reprezentację wymaganą przez aplikację odbierającą. Konwersja danych komunikatu aplikacji jest opcjonalna. Wymaga ona określenia opcji GMCONV w wywołaniu MQGET.

Opisywane są następujące elementy:

- Przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję GMCONV; patrz [“Przetwarzanie konwersji” na stronie 1466](#).

- Konwencje przetwarzania używane przez menedżera kolejek podczas przetwarzania wbudowanego formatu. Konwencje te są zalecane także dla programów zewnętrznych. Patrz sekcja [“Konwencje przetwarzania”](#) na stronie 1467.
- Specjalne uwagi dotyczące konwersji komunikatów raportu; patrz [“Konwersja komunikatów raportu”](#) na stronie 1471.
- Parametry przekazane do wyjścia konwersji danych (data-conversion exit); patrz [“MQCONVX-wyjście konwersji danych”](#) na stronie 1482.
- Wywołanie, którego można użyć z wyjścia w celu przekształcenia danych znakowych między różnymi reprezentacjami; patrz [“MQXCNV-Przekształć znaki”](#) na stronie 1477.
- Parametr struktury danych, który jest specyficzny dla wyjścia; patrz [“MQDXP-Dane-parametr wyjścia konwersji danych”](#) na stronie 1472.

Przetwarzanie konwersji

Te informacje opisują przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję GMCONV.

Menedżer kolejek wykonuje następujące działania, jeśli w wywołaniu MQGET określono opcję GMCONV i istnieje komunikat, który ma zostać zwrócony do aplikacji:

1. Jeśli co najmniej jedna z następujących wartości jest prawdziwa, konwersja nie jest konieczna:
 - Dane komunikatu znajdują się już w zestawie znaków i kodowaniu wymaganym przez aplikację, która wywołała wywołanie MQGET. Przed wywołaniem wywołania aplikacja musi ustawić pola *MDCSI* i *MDENC* w parametrze *MSGDSC* wywołania MQGET na wartości wymagane.
 - Długość danych komunikatu wynosi zero.
 - Długość parametru *BUFFER* wywołania MQGET wynosi zero.

W takich przypadkach komunikat jest zwracany bez konwersji do aplikacji wywołujących wywołanie MQGET; wartości *MDCSI* i *MDENC* w parametrze *MSGDSC* są ustawiane na wartości w informacjach sterujących w komunikacie, a wywołanie kończy się jedną z następujących kombinacji kodu zakończenia i kodu przyczyny:

Kod zakończenia

Kod przyczyny

CCOK

RCBRAK

CCWARN

RC2079

CCWARN

RC2080

Następujące kroki są wykonywane tylko wtedy, gdy zestaw znaków lub kodowanie danych komunikatu różni się od odpowiedniej wartości w parametrze *MSGDSC* i istnieją dane do przekształcenia:

1. Jeśli pole *MDFMT* w informacjach sterujących w komunikacie ma wartość FMNONE, to komunikat jest zwracany bez konwersji, z kodem zakończenia CCWARN i kodem przyczyny RC2110.
We wszystkich innych przypadkach przetwarzanie konwersji jest kontynuowane.
2. Komunikat zostanie usunięty z kolejki i umieszczony w tymczasowym buforze, który ma taką samą wielkość jak parametr *BUFFER*. W przypadku operacji przeglądania komunikat jest kopiowany do tymczasowego buforu, a nie do usunięcia z kolejki.
3. Jeśli komunikat ma zostać obciążony, aby zmieścić się w buforze, wykonaj następujące czynności:
 - Jeśli opcja GMATM *nie* została określona, zwracany jest komunikat bez konwersji, kod zakończenia CCWARN i kod przyczyny RC2080.
 - Jeśli opcja GMATM *była* jest określona, kod zakończenia jest ustawiany na CCWARN, kod przyczyny jest ustawiany na RC2079, a przetwarzanie konwersji będzie kontynuowane.

4. Jeśli komunikat może być zakwaterowany w buforze bez obcinania lub podano opcję GMATM, wykonaj następujące czynności:

- Jeśli format jest formatem wbudowanym, bufor jest przekazywany do usługi konwersji danych menedżera kolejek.
- Jeśli format nie jest formatem wbudowanym, bufor jest przekazywany do wyjścia napisanego przez użytkownika, które ma taką samą nazwę jak format. Jeśli nie można znaleźć wyjścia, zwracany jest komunikat bez konwersji, z kodem zakończenia CCWARN i kodem przyczyny RC2110.

Jeśli nie wystąpi błąd, dane wyjściowe z usługi konwersji danych lub z wyjścia napisanego przez użytkownika są komunikatem przekształconym, a kod zakończenia i kod przyczyny są zwracane do aplikacji wywołujących wywołanie MQGET.

5. Jeśli konwersja zakończy się pomyślnie, menedżer kolejek zwraca przekształcony komunikat do aplikacji. W tym przypadku kod zakończenia i kod przyczyny zwrócone przez wywołanie MQGET będą zazwyczaj jedną z następujących kombinacji:

Kod zakończenia

Kod przyczyny

CCOK

RCBRAK

CCWARN

RC2079

Jeśli jednak konwersja jest wykonywana przy użyciu wyjścia napisanego przez użytkownika, mogą zostać zwrócone inne kody przyczyny, nawet jeśli konwersja jest pomyślna.

Jeśli konwersja nie powiedzie się (bez względu na przyczynę), menedżer kolejek zwróci do aplikacji komunikat o nieprzekształconej wersji, a pola *MDCSI* i *MDENC* w parametrze *MSGDSC* ustawiają wartości w informacjach sterujących w komunikacie, a także kod zakończenia CCWARN.

Konwencje przetwarzania

Podczas przekształcania wbudowanego formatu menedżer kolejek jest zgodny z konwencjami przetwarzania opisanymi w tym temacie.

Należy rozważyć zastosowanie tych konwencji do programów zewnętrznych napisanych przez użytkownika, chociaż nie jest to wymuszane przez menedżer kolejek. Wbudowane formaty przekształcone przez menedżera kolejek to:

Wbudowane w formaty
FMADMN
FMMDE
FMCICS
FMPCF
FMCMD1
FMRMH
FMCMD2
FMRFH
FMDLH
FMRFH2
FMDH
FMSTR
FMEVNT

Wbudowane w formaty
FMTM
FMIMS
FMXQH
FMIMVS

1. Jeśli komunikat zostanie rozwinięty podczas konwersji, a jego wielkość przekracza wartość parametru *BUFFER*, zostanie wykonane następujące czynności:
 - Jeśli opcja *GMATM* nie została określona, zwracany jest komunikat bez konwersji, z kodem zakończenia *CCWARN* i kodem przyczyny *RC2120*.
 - Jeśli opcja *GMATM* was została określona, komunikat jest obcinany, kod zakończenia jest ustawiany na *CCWARN*, kod przyczyny jest ustawiony na *RC2079*, a przetwarzanie konwersji będzie kontynuowane.
2. Jeśli wystąpi obcięcie (przed lub podczas konwersji), to możliwe jest, że liczba poprawnych bajtów zwracanych w parametrze *BUFFER* będzie *mniej* niż długość buforu.

Może się to zdarzyć na przykład wtedy, gdy 4-bajtowa liczba całkowita lub znak DBCS jest znakiem końca buforu. Niekompletny element informacji nie jest przekształcany, a więc te bajty w zwróconym komunikacie nie zawierają poprawnych informacji. Może to również wystąpić, jeśli komunikat, który został obcięty przed konwersją, został obcięty podczas konwersji.

Jeśli liczba zwróconych poprawnych bajtów jest mniejsza niż długość buforu, nieużywane bajty na końcu buforu są ustawione na wartości *NULL*.
3. Jeśli tablica lub łańcuch znaków jest końcem buforu, to konwertowane jest tyle danych, ile jest to możliwe; tylko określony element tablicy lub znak DBCS, który jest niekompletny, nie jest przekształcany-przekształcane są poprzedzające je elementy tablicy lub znaki.
4. Jeśli wystąpi obcięcie (przed lub podczas konwersji), długość zwrócona dla parametru *DATLEN* jest długością komunikatu *unconverted* przed obcięciem.
5. Gdy łańcuchy są konwertowane między jednobajtowymi zestawami znaków (SBCS), dwubajtowymi zestawami znaków (DBCS) lub wielobajtowymi zestawami znaków (MBCS), łańcuchy mogą się rozszerzać lub zawierać kontrakt.
 - W formatach PCF: *FMADMN*, *FMEVNT* i *FMPCF*, łańcuchy w strukturach *MQCFST* i *MQCFSL* są rozszerzać lub kontrastować w razie potrzeby, aby pomieścić łańcuch po konwersji.

W przypadku struktury łańcuchowej *MQCFSL* łańcuchy znajdujące się na liście mogą rozszerzać się lub zawierać kontrakt o różne kwoty. W takim przypadku menedżer kolejek dopełnia krótsze łańcuchy zawierające odstępy, aby ich długość była taka sama jak najdłuższy łańcuch po konwersji.
 - W formacie *FMRMH* łańcuchy zaadresowane przez pola *RMSEO*, *RMSNO*, *RMDEO*i *RMDNO*, w zależności od potrzeb, w celu dostosowania do łańcuchów po konwersji lub w celu ich dostosowania.
 - W formacie *FMRFH* pole *RFNVS* rozwija lub kontrakty w razie potrzeby, aby pomieścić pary nazwa/wartość po konwersji.
 - W strukturach o stałych wielkościach pól menedżer kolejek zezwala na rozszerzanie lub kontrastowanie łańcuchów w obrębie ich pól stałych, o ile nie utracono istotnych informacji. W tym zakresie końcowe odstępy i znaki występujące po pierwszym znaku null w polu są traktowane jako nieistotne.
 - Jeśli łańcuch zostanie rozwinięty, ale tylko nieistotne znaki muszą zostać usunięte, aby pomieścić przekształcony łańcuch w polu, konwersja powiedzie się, a wywołanie kończy się z *CCOK* i kodem przyczyny *RCNONE* (przy założeniu, że nie wystąpiły inne błędy).
 - Jeśli łańcuch zostanie rozwinięty, ale przekształcony łańcuch wymaga usunięcia znaczących znaków, aby zmieścić się w tym polu, komunikat zostanie zwrócony bez konwersji, a wywołanie kończy się znakiem *CCWARN* i kodem przyczyny *RC2190*.

Uwaga: Kod przyczyny RC2190 powoduje, że w tym przypadku określono, czy określono opcję GMATM.

- W przypadku umów łańcuchowych menedżer kolejek dopełnia łańcuch odstępami na długość pola.

6. W przypadku komunikatów składających się z co najmniej jednej struktury nagłówka produktu IBM MQ, po której następuje dane użytkownika, możliwe jest przekształcenie jednej lub większej liczby struktur nagłówka, podczas gdy pozostała część komunikatu nie jest dostępna. Jednak z dwoma wyjątkami pola *MDCSI* i *MDENC* w każdej strukturze nagłówka zawsze poprawnie wskazują zestaw znaków i kodowanie danych, które są zgodne ze strukturą nagłówka.

Istnieją dwa wyjątki: struktury MQCIH i MQIIH, w których wartości w polach *MDCSI* i *MDENC* w tych strukturach nie są znaczące. W przypadku tych struktur dane, które są zgodne ze strukturą, znajdują się w tym samym zestawie znaków i kodowaniu, co sama struktura MQCIH lub MQIIH.

7. Jeśli pola *MDCSI* lub *MDENC* w informacjach sterujących komunikatu są pobierane lub w parametrze *MSGDSC*, określ wartości, które są niezdefiniowane lub nie są obsługiwane, menedżer kolejek może zignorować błąd, jeśli niezdefiniowana lub nieobsługiwana wartość nie musi być używana podczas przekształcania komunikatu.

Na przykład, jeśli pole *MDENC* w komunikacie określa nieobsługiwane kodowanie zmiennopozycyjne, ale komunikat zawiera tylko dane będące liczbami całkowitymi lub zawiera dane zmiennopozycyjne, które nie wymagają konwersji (ponieważ kodowanie źródłowe i docelowe zmiennopozycyjne są identyczne), błąd może lub nie jest diagnozowany.

Jeśli błąd jest diagnozowany, zwracany jest komunikat bez konwersji, kod zakończenia CCWARN i jeden z kodów przyczyny RC2111, RC2112, RC2113, RC2114 lub RC2115, RC2116, RC2117, RC2118 (odpowiednio); pola *MDCSI* i *MDENC* w parametrze *MSGDSC* są ustawiane na wartości w informacjach sterujących w komunikacie.

Jeśli błąd nie zostanie wykryty, a konwersja zakończy się pomyślnie, wartości zwrócone w polach *MDCSI* i *MDENC* w parametrze *MSGDSC* są wartościami określonymi przez aplikację wywołującą wywołanie MQGET.

8. We wszystkich przypadkach, jeśli komunikat jest zwracany do aplikacji bez konwersji, kod zakończenia jest ustawiany na CCWARN, a pola *MDCSI* i *MDENC* w parametrze *MSGDSC* są ustawione na wartości odpowiednie dla danych, które nie zostały przekształcone. Jest to wykonywane również dla FMNONE.

Parametr *REASON* jest ustawiony na kod wskazujący, dlaczego nie można było wykonać konwersji, chyba że komunikat musiał zostać obcięty. Kody przyczyny związane z obcięciem mają pierwszeństwo przed kodami przyczyny związanymi z konwersją. (Aby określić, czy obcięty komunikat został przekształcony, należy sprawdzić wartości zwracane w polach *MDCSI* i *MDENC* w parametrze *MSGDSC*).

W przypadku zdiagnozowania błędu zwracany jest konkretny kod przyczyny lub ogólny kod przyczyny RC2119. Zwrócony kod przyczyny zależy od możliwości diagnostycznych bazowej usługi konwersji danych.

9. Jeśli kod zakończenia CCWARN jest zwracany, a istotny jest więcej niż jeden kod przyczyny, kolejność wykonywania operacji jest następująca:

- a. Następujące przyczyny mają pierwszeństwo przed wszystkimi innymi:

- RC2079

- b. Następny z kolei ma następującą przyczynę:

- RC2110

- c. Kolejność wykonywania operacji w pozostałych kodach przyczyny nie jest zdefiniowana.

10. Po zakończeniu wywołania MQGET:

- Następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:

- RCBRAK

- Następujący kod przyczyny wskazuje, że komunikat *może* zostać pomyślnie przekształcony (należy sprawdzić pola *MDCSI* i *MDENC* w parametrze *MSGDSC*, aby dowiedzieć się):
 - RC2079
- Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Następujące przetwarzanie jest specyficzne dla formatów wbudowanych; nie ma zastosowania do formatów zdefiniowanych przez użytkownika:

1. Z wyjątkiem następujących formatów:

- FMADMN
- FMEVNT
- FMIMVS
- FMPCF
- FMSTR

żaden z wbudowanych formatów nie może być konwertowany z lub do zestawów znaków, które nie mają znaków SBCS dla znaków, które są poprawne w nazwach kolejek. Jeśli podejmowana jest próba wykonania takiej konwersji, komunikat jest zwracany bez konwersji, z kodem zakończenia CCWARN i kodem przyczyny RC2111 lub RC2115, w zależności od przypadku.

Zestaw znaków Unicode UCS-2 jest przykładem zestawu znaków, który nie ma znaków SBCS dla znaków, które są poprawne w nazwach kolejek.

2. Jeśli dane komunikatu dla wbudowanego formatu są obcinane, pola w komunikacie zawierające długości łańcuchów lub liczby elementów lub struktur *nie* są dostosowywane w taki sposób, aby odzwierciedlały długość danych zwracanych do aplikacji; wartości zwracane dla takich pól w danych komunikatu są wartościami mającymi zastosowanie do komunikatu *przed obcięciem*.

Podczas przetwarzania komunikatów, takich jak obcięty komunikat FMADMN, należy zwrócić uwagę na to, aby aplikacja nie próbowała uzyskać dostępu do danych znajdujących się poza końcem zwróconych danych.

3. Jeśli nazwa formatu to FMDLH, dane komunikatu zaczynają się od struktury MQDLH, po czym może następować zero lub większa liczba bajtów danych komunikatu aplikacji. Format, zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane za pomocą pól *DLFMT*, *DLCSI* i *DLENC* w strukturze MQDLH na początku komunikatu. Ponieważ struktura MQDLH i dane komunikatu aplikacji mogą mieć różne zestawy znaków i kodowania, możliwe jest, aby konwersja była jedna, druga lub obie struktury MQDLH, a także dane komunikatu aplikacji.

Menedżer kolejek przekształca najpierw strukturę MQDLH w razie potrzeby. Jeśli konwersja zakończy się pomyślnie, lub struktura MQDLH nie wymaga konwersji, menedżer kolejek sprawdza pola *DLCSI* i *DLENC* w strukturze MQDLH, aby sprawdzić, czy konwersja danych komunikatu aplikacji jest wymagana. Jeśli wymagana jest konwersja *jest* wymagana, menedżer kolejek wywołuje wyjście napisane przez użytkownika z nazwą nadaną przez pole *DLFMT* w strukturze MQDLH lub wykonuje samą konwersję (jeśli *DLFMT* jest nazwą wbudowanego formatu).

Jeśli wywołanie MQGET zwróci kod zakończenia CCWARN, a kod przyczyny jest jednym z tych, które wskazują, że konwersja nie powiodła się, zastosowanie ma jedna z następujących sytuacji:

- Nie można przekształcić struktury MQDLH. W tym przypadku dane komunikatu aplikacji nie zostaną przekształcone.
- Struktura MQDLH została przekształcona, ale dane komunikatu aplikacji nie zostały przekształcone.

Aplikacja może sprawdzić wartości zwracane w polach *MDCSI* i *MDENC* w parametrze *MSGDSC*, a także wartości w strukturze MQDLH, aby określić, które z poprzednich zastosowań mają zastosowanie.

4. Jeśli nazwą formatu jest FMXQH, dane komunikatu zaczynają się od struktury MQXQH, po czym może następować zero lub większa liczba bajtów dodatkowych danych. Te dodatkowe dane są zwykle danymi komunikatu aplikacji (może to być zerowa długość), ale może być również jeden lub więcej dalszych struktur nagłówka IBM MQ na początku dodatkowych danych.

Struktura MQXQH musi znajdować się w zestawie znaków i kodowaniu menedżera kolejek. Format, zestaw znaków i kodowanie danych zgodnie ze strukturą MQXQH są nadawane przez pola *MDFMT*, *MDCSI* i *MDENC* w strukturze MQMD, które zawierają w zmaterializowanych tabelach MQXQH. Dla każdej kolejnej struktury nagłówek IBM MQ, pola *MDFMT*, *MDCSI* i *MDENC* w strukturze opisują dane, które są zgodne z tą strukturą. Dane te są albo inną strukturą nagłówek IBM MQ, albo danymi komunikatu aplikacji.

Jeśli dla komunikatu FMXQH określono opcję GMCONV, dane komunikatu aplikacji i niektóre struktury nagłówek MQ są przekształcane, *ale dane w strukturze MQXQH nie są danymi*. Z tego powodu w przypadku powrotu z wywołania MQGET:

- Wartości pól *MDFMT*, *MDCSI* i *MDENC* w parametrze *MSGDSC* opisują dane w strukturze MQXQH, a *nie* dane komunikatu aplikacji. Wartości *nie* będą takie same jak wartości określone przez aplikację, która wygenerował wywołanie MQGET.

Wynika to z tego, że aplikacja, która wielokrotnie pobiera komunikaty z kolejki transmisji z określoną opcją GMCONV, musi zresetować pola *MDCSI* i *MDENC* w parametrze *MSGDSC* do wartości niezbędnych dla danych komunikatu aplikacji, przed każdym wywołaniem MQGET.

- Wartości pól *MDFMT*, *MDCSI* i *MDENC* w ostatniej strukturze nagłówek MQ zawierają opis danych komunikatu aplikacji. Jeśli nie istnieją inne struktury nagłówek produktu IBM MQ, dane komunikatu aplikacji są opisywane przez te pola w strukturze MQMD w strukturze MQXQH. Jeśli konwersja powiedzie się, wartości będą takie same jak wartości określone w parametrze *MSGDSC* przez aplikację, która wywołała wywołanie MQGET.

Jeśli komunikat jest komunikatem o rozdzielaniu, w strukturze MQXQH następuje struktura MQDH (wraz z tablicami rekordów MQOR i MQPMR), po czym po kolei mogą następować zero lub więcej struktur nagłówek IBM MQ oraz zero lub więcej bajtów danych komunikatu aplikacji. Podobnie jak struktura MQXQH, struktura MQDH musi znajdować się w zestawie znaków i kodowaniu menedżera kolejek, a nie jest ona konwertowana w wywołaniu MQGET, nawet jeśli określono opcję GMCONV.

Przetwarzanie opisanych wcześniej struktur MQXQH i MQDH jest przeznaczone przede wszystkim do użycia przez agenty kanału komunikatów podczas pobierania komunikatów z kolejki transmisji.

Konwersja komunikatów raportu

Komunikat raportu może zawierać różne ilości danych komunikatu aplikacji, zgodnie z opcjami raportu określonymi przez nadawcę oryginalnego komunikatu.

W szczególności komunikat raportu może zawierać:

1. Brak danych komunikatu aplikacji
2. Niektóre z danych komunikatu aplikacji z oryginalnego komunikatu

Dzieje się tak wtedy, gdy nadawca oryginalnego komunikatu określa RO* D, a komunikat jest dłuższy niż 100 bajtów.

3. Wszystkie dane komunikatu aplikacji z oryginalnego komunikatu

Dzieje się tak wtedy, gdy nadawca oryginalnego komunikatu określa RO* F, lub określa RO* D, a komunikat ma 100 bajtów lub krótszy.

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, kopiuje on nazwę formatu z oryginalnego komunikatu do pola *MDFMT* w informacjach sterujących w komunikacie raportu. Nazwa formatu w komunikacie raportu może więc oznaczać, że długość danych różni się od długości podanej w komunikacie raportu (opisywane wcześniej przypadki 1 i 2).

Jeśli opcja GMCONV jest określona podczas pobierania komunikatu raportu:

- Dla przypadku 1 opisanego poprzednio, wyjście konwersji danych nie zostanie wywołane (ponieważ komunikat raportu nie będzie miał danych).
- W przypadku opisanego wcześniej przypadku 3 nazwa formatu musi być poprawna, co oznacza długość danych komunikatu.

- Jednak w przypadku opisywanego wcześniej przypadku 2 wyjście konwersji danych zostanie wywołane w celu przekształcenia komunikatu, który jest *krótszy*, niż długość wynikający z nazwy formatu.

Dodatkowo kodem przyczyny przekazanej do wyjścia będzie zazwyczaj RCNONE (czyli kod przyczyny nie wskazuje, że komunikat został obcięty). Dzieje się tak dlatego, że dane komunikatu zostały obcięte przez *nadawcę* komunikatu raportu, a nie przez menedżer kolejek odbiorcy w odpowiedzi na wywołanie MQGET.

Ze względu na te możliwości wyjście konwersji danych *nie powinno* używać nazwy formatu do odliczenia długości przekazywanych do niej danych; zamiast tego wyjście powinno sprawdzać długość podanych danych i być przygotowane do konwersji danych *mniej* niż długość implikowana przez nazwę formatu. Jeśli dane mogą zostać przekształcone pomyślnie, kod zakończenia CCOK i kod przyczyny RCNONE powinny zostać zwrócone przez wyjście. Długość danych komunikatu, które mają zostać przekształcone, jest przekazywana do wyjścia jako parametr *INLEN*.

Interfejs programistyczny wrażliwy na produkt

Jeśli komunikat raportu zawiera informacje na temat działania, które miało miejsce, jest ono znane jako raport aktywności. Przykładami działań są:

- Agent MCA wysyłający komunikat z kolejki w dół kanału
- Agent MCA odbierający komunikat z kanału i umieszczający go w kolejce
- Niedostarczalny komunikat MCA w kolejkowaniu niedostarczanego komunikatu
- Agent MCA pobierający komunikat z kolejki i odrzucający go
- Program obsługi niedostarczonych komunikatów umieszczający komunikat z powrotem w kolejce
- Serwer komend przetwarzający żądanie PCF-broker przetwarzający żądanie publikowania
- Aplikacja użytkownika pobierający komunikat z kolejki-aplikacja użytkownika przeglądający komunikat w kolejce

Każda aplikacja, w tym menedżer kolejek, może dodać niektóre dane komunikatu do raportu działań po nagłówku raportu. Ilość danych, które powinny być dostarczone, jeśli część jest wysyłana, nie jest stała i jest rozstrzygana przez aplikację. Zwrócone informacje powinny być użyteczne dla aplikacji przetwarzając raport działań. Raporty aktywności menedżera kolejek zwrócą wraz z nimi wszystkie standardowe struktury nagłówka IBM MQ (rozpoczynające się 'MQH') zawarte w oryginalnym komunikacie. Obejmuje to na przykład wszystkie nagłówki MQRFH2, które zostały dołączone do oryginalnego komunikatu. Ponadto menedżer kolejek zwróci nagłówek MQCFH, ale nie są powiązane z nim parametry PCF. Dzięki temu aplikacje monitorujące są pomysłowe na to, o czym była wiadomość.

MQDXP-Dane-parametr wyjścia konwersji danych

Blok parametru wyjścia konwersji danych.

Przegląd

Cel: Struktura MQDXP jest parametrem, który jest przekazywany przez menedżer kolejek do wyjścia konwersji danych po wywołaniu wyjścia w celu przekształcenia danych komunikatu w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat wyjścia konwersji danych można znaleźć w opisie wywołania MQCONVX.

Zestaw znaków i kodowanie: Dane znakowe w produkcie MQDXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Dane te są nadawane przez atrybut menedżera kolejek produktu *CodedCharSetId*. Dane numeryczne w MQDXP znajdują się w rodzimym kodowaniu komputera. Dane te są nadawane przez ENNAT.

Użycie: Tylko pola *DXLEN*, *DXCC*, *DXREA* i *DXRES* w produkcie MQDXP mogą zostać zmienione przez wyjście; zmiany w innych polach są ignorowane. Jednak pole *DXLEN* *nie może* zostać zmienione, jeśli przekształcany komunikat jest to segment, który zawiera tylko część komunikatu logicznego.

Gdy sterowanie powraca do menedżera kolejek z wyjścia, menedżer kolejek sprawdza wartości zwrócone w MQDXP. Jeśli zwrócone wartości nie są poprawne, menedżer kolejek kontynuuje przetwarzanie, tak jakby program zewnętrzny zwrócił wartość XRFAIL w produkcie DXRES. Jednak menedżer kolejek ignoruje wartości pól DXCC i DXREA zwracanych przez wyjście w tym przypadku, a zamiast tych wartości te pola miały wartość input dla wyjścia. Następujące wartości w tabeli MQDXP powodują, że przetwarzanie to ma miejsce:

- Pole DXRES nie jest polem XROK, a nie XRFAIL
- Pole DXCC nie jest CCOK, a nie CCWARN
- Pole DXLEN mniejsze niż zero lub pole DXLEN zmienione, gdy przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.
- [“Pola” na stronie 1473](#)
- [“Deklaracja RPG \(plik kopii CMQDXPH\)” na stronie 1477](#)

Pola

Struktura MQDXP zawiera następujące pola: pola są opisane w **porządku alfabetycznym**:

DXAOP (10-cyfrowa liczba całkowita ze znakiem)

Opcje aplikacji.

To jest kopia pola GMOPT struktury MQGMO określonej przez aplikację wywołującą wywołanie MQGET. Wyjście może być konieczne, aby sprawdzić, czy określono opcję GMATM.

To jest pole wejściowe do wyjścia.

DXCC (10-cyfrowa liczba całkowita ze znakiem)

Kod zakończenia.

Po wywołaniu wyjścia zawiera on kod zakończenia, który zostanie zwrócony do aplikacji, która wywołała wywołanie MQGET, jeśli program obsługi wyjścia nie zdecyduje się na nic. Zawsze jest to CCWARN, ponieważ albo wiadomość została obcięta, albo komunikat wymaga konwersji, a to jeszcze nie zostało zrobione.

Po wyjściu z wyjścia to pole zawiera kod zakończenia, który ma zostać zwrócony do aplikacji w parametrze CMPCOD wywołania MQGET. Poprawne są tylko wartości CCOK i CCWARN. Aby uzyskać sugestie dotyczące sposobu, w jaki wyjście powinno ustawić to pole na wyjściu, należy zapoznać się z opisem pola DXREA.

Jest to pole wejściowe/wyjściowe do wyjścia.

DXCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków wymagany przez aplikację.

Jest to identyfikator kodowanego zestawu znaków zestawu znaków wymagany przez aplikację wywołującą wywołanie MQGET. Więcej informacji zawiera pole MDCSI w strukturze MQMD. Jeśli aplikacja określi wartość specjalną CSQM w wywołaniu MQGET, menedżer kolejek zmieni to na rzeczywisty identyfikator zestawu znaków zestawu znaków używanego przez menedżer kolejek przed wywołaniem wyjścia.

Jeśli konwersja zakończy się pomyślnie, wyjście powinno skopiować to pole do pola MDCSI w deskrypcorze komunikatu.

To jest pole wejściowe do wyjścia.

DXENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie numeryczne wymagane przez aplikację.

Jest to kodowanie liczbowe, które jest wymagane przez aplikację wywołującą wywołanie MQGET. Więcej szczegółów zawiera pole MDENC w strukturze MQMD.

Jeśli konwersja zakończy się pomyślnie, wyjście powinno skopiować to pole do pola *MDENC* w deskrytorze komunikatu.

To jest pole wejściowe do wyjścia.

DXHCN (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt połączenia.

Jest to uchwyt połączenia, który może być używany w wywołaniu MQXCNCV. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

DXLEN (10-cyfrowa liczba całkowita ze znakiem)

Długość (w bajtach) danych komunikatu.

Po wywołaniu wyjścia w tym polu znajduje się oryginalna długość danych komunikatu aplikacji. Jeśli komunikat został obcięty, aby zmieścić się w buforze udostępnionym przez aplikację, wielkość komunikatu udostępnionego do wyjścia będzie *mniejsza* niż wartość *DXLEN*. Wielkość komunikatu dostarczanego do wyjścia jest zawsze nadawana przez parametr *INLEN* wyjścia, niezależnie od tego, czy nastąpiło obcięcie, jakie mogło wystąpić.

Obcięcie jest wskazywane przez pole *DXREA* o wartości RC2079 na wejściu do wyjścia.

Większość konwersji nie będzie wymagać zmiany tej długości, ale wyjście może to zrobić w razie potrzeby; wartość ustawiona przez wyjście jest zwracana do aplikacji w parametrze *DATLEN* wywołania MQGET. Jednak ta długość *nie może* zostać zmieniona, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego. Jest to spowodowane tym, że zmiana długości spowoduje, że przesunięcia późniejszych segmentów w komunikacie logicznym są niepoprawne.

Należy zwrócić uwagę, że jeśli wyjście chce zmienić długość danych, należy pamiętać, że menedżer kolejek już zdecydował, czy dane komunikatu będą pasowały do buforu aplikacji, na podstawie długości danych *nieprzekształconych*. Ta decyzja określa, czy komunikat jest usuwany z kolejki (lub przeniesiono kursor przeglądania, dla żądania przeglądania) i nie ma wpływu na zmianę długości danych spowodowaną przez konwersję. Z tego powodu zaleca się, aby wyjścia konwersji nie powodował zmiany długości danych komunikatu aplikacji.

Jeśli konwersja znaków oznacza zmianę długości, łańcuch może zostać przekształcony w inny łańcuch o tej samej długości w bajtach, obcinanie odstępów końcowych lub dopełnianie odstępami w razie potrzeby.

Wyjście nie jest wywoływane, jeśli komunikat nie zawiera danych komunikatu aplikacji, dlatego wartość *DXLEN* jest zawsze większa od zera.

Jest to pole wejściowe/wyjściowe do wyjścia.

DXREA (10-cyfrowa liczba całkowita ze znakiem)

Kod przyczyny kwalifikujący *DXCC*.

Po wywołaniu wyjścia zawiera on kod przyczyny, który zostanie zwrócony do aplikacji, która wywołała wywołanie MQGET, jeśli program obsługi wyjścia nie zdecyduje się na nic. Wśród możliwych wartości znajdują się wartości RC2079, co oznacza, że komunikat został obcięty w celu dopasowania do buforu udostępnionego przez aplikację, oraz RC2119, co oznacza, że komunikat wymaga konwersji, ale nie zostało to jeszcze wykonane.

W przypadku wyjścia z wyjścia to pole zawiera przyczynę, która ma zostać zwrócona do aplikacji w parametrze *REASON* wywołania MQGET. Zalecane jest następujące ustawienie:

- Jeśli parametr *DXREA* miał wartość RC2079 na wejściu do wyjścia, pola *DXREA* i *DXCC* nie powinny być zmieniane, bez względu na to, czy konwersja powiodła się, czy też nie.

(Jeśli pole *DXCC* nie jest polem CCOK, aplikacja, która pobiera komunikat, może zidentyfikować błąd konwersji, porównując zwrócone wartości *MDENC* i *MDCSI* w deskrytorze komunikatu z żądanymi wartościami; w przeciwnym razie aplikacja nie może odróżnić obciętej wiadomości od

komunikatu, który właśnie dopasował bufor. Z tego powodu RC2079 należy zwrócić w preferowanej kolejności do któregośkolwiek z powodów wskazujących na niepowodzenie konwersji.)

- Jeśli program *DXREA* ma jakąkolwiek inną wartość na wejściu do wyjścia:

- Jeśli konwersja powiedzie się, parametr *DXCC* powinien zostać ustawiony na wartość CCOK, a parametr *DXREA* ustawiony na wartość RCNONE.
- Jeśli konwersja nie powiedzie się lub komunikat zostanie rozwinięty i musi zostać obcięty, aby zmieścić się w buforze, wartość *DXCC* powinna zostać ustawiona na CCWARN (lub pozostaw bez zmian), a parametr *DXREA* na jedną z następujących wartości i na poniższej liście, aby wskazać rodzaj niepowodzenia.

Należy zwrócić uwagę, że jeśli komunikat po konwersji jest zbyt duży dla buforu, powinien on zostać obcięty tylko wtedy, gdy aplikacja, która wydała wywołanie MQGET, określiła opcję GMATM:

- Jeśli ta opcja została określona, należy zwrócić przyczynę RC2079 .
- Jeśli ta opcja nie została określona, komunikat powinien zostać zwrócony bez konwersji z kodem przyczyny RC2120.

Kody przyczyny znajdujące się na poniższej liście są zalecane do użycia przez wyjście w celu wskazania przyczyny niepowodzenia konwersji, ale wyjście może zwrócić inne wartości z zestawu kodów RC*, jeśli jest to uważane za odpowiednie. Ponadto zakres wartości od RC0900 do RC0999 jest przydzielany do użycia przez wyjście w celu wskazania warunków, które program obsługi wyjścia chce komunikować z aplikacją wywołującym wywołanie MQGET.

Uwaga: Jeśli komunikat nie może zostać pomyślnie przekształcony, wyjście *musi* zwrócić wartość XRFAIL w polu *DXRES* , aby menedżer kolejek mógł zwrócić nieskonwertowany komunikat. Jest to prawda, niezależnie od kodu przyczyny zwróconego w polu *DXREA* .

RC0900

(900, X'384 ') Najniższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

RC0999

(999, X'3E7') Najwyższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

RC2120

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

RC2119

(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

RC2111

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

RC2113

(2113, X'841 ') Zpakowane kodowanie dziesiętne w komunikacie nie zostało rozpoznane.

RC2114

(2114, X'842 ') Kodowanie zmiennopozycyjne w komunikacie nie zostało rozpoznane.

RC2112

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

RC2115

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

RC2117

(2117, X'845 ') Zpakowane-kodowanie dziesiętne określone przez odbiornik nierozpoznany.

RC2118

(2118, X'846 ') Kodowanie zmiennopozycyjne określone przez odbiornik nie jest rozpoznawane.

RC2116

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

RC2079

(2079, X'81F') Zwrócona została obcięta wiadomość (przetwarzanie zostało zakończone).

Jest to pole wejściowe/wyjściowe do wyjścia.

DXRES (10-cyfrowa liczba całkowita ze znakiem)

Odpowiedź z wyjścia.

Wartość ta jest ustawiana przez wyjście w celu wskazania powodzenia lub innej konwersji. Musi to być jeden z następujących elementów:

XROK

Konwersja powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca następujące informacje do aplikacji, która wywołała wywołanie MQGET:

- Wartość pola *DXCC* na wyjściu z wyjścia
- Wartość pola *DXREA* na wyjściu z wyjścia
- Wartość pola *DXLEN* na wyjściu z wyjścia
- Zawartość buforu wyjściowego wyjścia *OUTBUF*. Liczba zwróconych bajtów jest mniejsza od parametru *OUTLEN* wyjścia, a wartość pola *DXLEN* na wyjściu z wyjścia

Jeśli pola *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia mają wartość *both* bez zmian, menedżer kolejek zwraca:

- Wartość pól *MDENC* i *MDCSI* w strukturze MQDXP w *danych wejściowych* do wyjścia.

Jeśli jeden lub oba pola *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia zostały zmienione, menedżer kolejek zwraca następujące dane:

- Wartość pól *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia na wyjściu wyjścia z wyjścia.
-

XRFAIL

Konwersja nie powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca następujące informacje do aplikacji, która wywołała wywołanie MQGET:

- Wartość pola *DXCC* na wyjściu z wyjścia
- Wartość pola *DXREA* na wyjściu z wyjścia
- Wartość pola *DXLEN* w *danych wejściowych* do wyjścia
- Zawartość buforu wejściowego wyjścia *INBUF*. Liczba zwróconych bajtów jest podawana przez parametr *INLEN*.

Jeśli wyjście zostało zmienione *INBUF*, wyniki są niezdefiniowane.

DXRES to pole wyjściowe z wyjścia.

DXSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DXSIDV

Identyfikator struktury parametru wyjścia konwersji danych.

To jest pole wejściowe do wyjścia.

DXVER (dziesięciocyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DXVER1

Numer wersji struktury parametru wyjścia konwersji danych.

Następująca stała określa numer wersji bieżącej wersji:

DXVERC

Bieżąca wersja struktury parametru wyjścia konwersji danych.

Uwaga: Gdy zostanie wprowadzona nowa wersja tej struktury, układ istniejącej części nie jest zmieniany. Wyjście powinno więc sprawdzić, czy pole *DXVER* jest równe lub większe od najniższej wersji, która zawiera pola, które ma być używane przez program obsługi wyjścia.

To jest pole wejściowe do wyjścia.

DXXOP (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartością jest 0.

Deklaracja RPG (plik kopii CMQDXPH)

```
D*..1....:....2.....3.....4.....5.....6.....7..
D* MQDXP Structure
D*
D* Structure identifier
D DXSID 1 4
D* Structure version number
D DXVER 5 8I 0
D* Reserved
D DXXOP 9 12I 0
D* Application options
D DXAOP 13 16I 0
D* Numeric encoding required by application
D DXENC 17 20I 0
D* Character set required by application
D DXCSI 21 24I 0
D* Length in bytes of message data
D DXLEN 25 28I 0
D* Completion code
D DXCC 29 32I 0
D* Reason code qualifying DXCC
D DXREA 33 36I 0
D* Response from exit
D DXRES 37 40I 0
D* Connection handle
D DXHCN 41 44I 0
```

MQXCNCV-Przekształć znaki

Wywołanie MQXCNCV konwertuje znaki z jednego zestawu znaków na inny.

To wywołanie jest częścią interfejsu DCI (Data Conversion Interface) produktu IBM MQ, który jest jednym z interfejsów środowiska produktu IBM MQ. Uwaga: To wywołanie może być używane tylko z wyjścia konwersji danych.

- [“Składnia” na stronie 1477](#)
- [“Parametry” na stronie 1478](#)
- [“Wywołanie RPG \(ILE\)” na stronie 1482](#)

Składnia

MQXCNCV (HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSI, TGTLEN, TGTBUF, DATLEN, CMPCOD, REASON)

Parametry

Wywołanie MQXCNCV ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Zwykle jest to uchwyt przekazany do wyjścia konwersji danych w polu *DXHCN* struktury MQDXP. Uchwyt ten nie musi być taki sam jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

W systemie IBM można określić następującą wartość specjalną dla *HCONN*:

HCDEFH

Domyślny uchwyt połączenia.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje, które sterują działaniem MQXCNCV.

Można podać zero lub więcej opcji opisanych w dalszej części tej sekcji. Jeśli wymagane jest więcej niż jedno, możliwe jest dodanie wartości (nie należy dodawać tej samej stałej więcej niż raz).

Domyślna-opcja konwersji: Poniższa opcja steruje użyciem domyślnej konwersji znaków:

DCCDEF

Konwersja domyślna.

Ta opcja określa, że domyślna konwersja znaków może być używana, jeśli jeden lub oba zestawy znaków określone w wywołaniu nie są obsługiwane. Umożliwia to menedżerowi kolejek korzystanie z domyślnego zestawu znaków określonego przez instalację, który przybliża określony zestaw znaków podczas przekształcania łańcucha.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji łańcucha jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Można tego uniknąć, używając w łańcuchu tylko znaków, które są wspólne zarówno dla określonego zestawu znaków, jak i domyślnego zestawu znaków.

Domyślne zestawy znaków są definiowane przy użyciu opcji konfiguracyjnej, gdy menedżer kolejek jest zainstalowany lub zrestartowany.

Jeśli parametr DCCDEF nie zostanie określony, menedżer kolejek używa tylko określonych zestawów znaków w celu przekształcenia łańcucha, a wywołanie nie powiedzie się, jeśli jeden lub oba zestawy znaków nie są obsługiwane.

Opcja dopełnienia: Poniższa opcja umożliwia menedżerowi kolejek dopełnianie przekształconego łańcucha za pomocą odstępów lub odrzucania nieistotnych znaków końcowych, tak aby przekształcony łańcuch pasował do buforu docelowego:

DCCFIL

Bufor docelowy wypełnienia.

Ta opcja wymaga, aby konwersja była wypełniona w taki sposób, aby bufor docelowy został całkowicie wypełniony:

- Jeśli po przekształceniu kontrakty łańcuchowe są przekształcane, to w celu wypełnienia bufora docelowego dodawane są odstępy końcowe.
- Jeśli łańcuch zostanie rozwinięty po przekształceniu, znaki końcowe, które nie są znaczące, zostaną odrzucone, aby przekształcony łańcuch pasował do bufora docelowego. Jeśli to działanie może zostać wykonane pomyślnie, wywołanie zakończy się z kodem powrotu CCOK i kodem przyczyny RCNONE.

Jeśli w buforze docelowym znajduje się zbyt mało znaczących znaków końcowych, to w buforze docelowym znajduje się dużo łańcucha, który będzie pasował, a wywołanie kończy się znakiem CCWARN i kodem przyczyny RC2120.

Nieistotne znaki to:

- Odstępy końcowe
- Znaki następujące po pierwszym znaku o kodzie zero w łańcuchu (ale z wyjątkiem pierwszego znaku o kodzie zero)
- Jeśli łańcuch, *TGTCSI* i *TGTLEN* są takie, że bufor docelowy nie może być całkowicie ustawiony z poprawnymi znakami, wywołanie kończy się niepowodzeniem z kodem CCFAIL i kodem przyczyny RC2144. Może się tak zdarzyć, gdy *TGTCSI* ma ustawiony zestaw znaków DBCS (na przykład UCS-2), ale *TGTLEN* określa długość nieparzystą (w bajtach).
- Wartość *TGTLEN* może być mniejsza lub większa niż *SRCLLEN*. W przypadku powrotu z tabeli MQXCNCV, produkt *DATLEN* ma taką samą wartość, jak *TGTLEN*.

Jeśli ta opcja nie jest określona:

- W razie potrzeby łańcuch może zostać zamówiony lub rozwinięty w buforze docelowym. Nieistotne znaki końcowe nie są dodawane ani usuwane.

Jeśli przekształcony łańcuch mieści się w buforze docelowym, wywołanie kończy się łańcuchem CCOK i kodem przyczyny RCNONE.

Jeśli przekształcony łańcuch jest zbyt duży dla buforu docelowego, tyle łańcucha, jaki będzie pasował, znajduje się w buforze docelowym, a wywołanie kończy się z CCWARN i kodem przyczyny RC2120. Należy zwrócić uwagę, że w tym przypadku może zostać zwróconych mniej niż *TGTLEN* bajtów.

- Wartość *TGTLEN* może być mniejsza lub większa niż *SRCLLEN*. W przypadku powrotu z MQXCNCV, *DATLEN* jest mniejsze lub równe *TGTLEN*.

Opcje kodowania: W celu określenia kodowania liczb całkowitych w łańcuchach źródłowych i docelowych można użyć następujących opcji. Odpowiednie kodowanie jest używane *tylko*, gdy odpowiedni identyfikator zestawu znaków wskazuje, że reprezentacja zestawu znaków w pamięci głównej jest zależna od kodowania używanego dla binarnych liczb całkowitych. Ma to wpływ tylko na niektóre wielobajtowe zestawy znaków (dla zestawów znaków example, UCS-2).

Kodowanie jest ignorowane, jeśli zestaw znaków to zestaw znaków jednobajtowych (SBCS) lub zestaw znaków wielobajtowych z reprezentacją w głównej pamięci masowej, która nie jest zależna od kodowania liczb całkowitych.

Należy określić tylko jedną z wartości DCCS*, w połączeniu z jedną z wartości DCCT*:

DCCSNA

Kodowanie źródłowe jest domyślne dla środowiska i języka programowania.

DCCSNO

Kodowanie źródłowe jest normalne.

DCCSRE

Kodowanie źródłowe zostało odwrócone.

DCCSUN

Kodowanie źródłowe jest niezdefiniowane.

DCCTNA

Kodowanie docelowe jest wartością domyślną dla środowiska i języka programowania.

DCCTNO

Kodowanie docelowe jest normalne.

DCCTRE

Kodowanie docelowe jest odwrócone.

DCCTUN

Kodowanie docelowe jest niezdefiniowane.

Zdefiniowane wcześniej wartości kodowania można dodać bezpośrednio do pola *OPTS*. Jeśli jednak kodowanie źródłowe lub docelowe jest uzyskiwane z pola *MDENC* w strukturze MQMD lub innej strukturze, należy wykonać następujące przetwarzanie:

1. Kodowanie liczb całkowitych musi zostać wyodrębnione z pola *MDENC* , eliminując kodowanie zmiennopozycyjne i upakowane dziesiętne. Więcej informacji na temat tego sposobu można znaleźć w sekcji “Analizowanie kodowania” na stronie 1461 .
2. Kodowanie całkowitoliczbowe wynikające z kroku 1 musi zostać pomnożone przez odpowiedni współczynnik zanim zostanie dodane do pola *OPTS* . Są to następujące czynniki:

DCCSFA

Współczynnik kodowania źródłowego

DCCTFA

Współczynnik dla kodowania docelowego

Jeśli nie zostanie podana, opcje kodowania zostaną domyślnie zdefiniowane jako niezdefiniowane (DCC* UN). W większości przypadków nie ma to wpływu na pomyślne zakończenie wywołania MQXCNCV. Jeśli jednak odpowiedni zestaw znaków to zestaw znaków wielobajtowych z reprezentacją zależną od kodowania (na przykład zestaw znaków UCS-2), wywołanie nie powiedzie się i odpowiednio kod przyczyny RC2112 lub RC2116 .

Opcja domyślna: Jeśli żadna z opcji opisanych wcześniej nie jest określona, można użyć następującej opcji:

DCCNON

Nie określono żadnych opcji.

DCCNON jest zdefiniowane w dokumentacji programu pomocy. Ta opcja nie jest przeznaczona do użycia z innymi, ale ponieważ jej wartość jest równa zero, nie można wykryć takiego użycia.

SRCCSI (10-cyfrowa liczba całkowita ze znakiem)-wejście

Identyfikator kodowanego zestawu znaków łańcucha przed konwersją.

Jest to identyfikator kodowanego zestawu znaków łańcucha wejściowego w produkcie *SRCBUF*.

SRCLen (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość łańcucha przed konwersją.

Jest to długość w bajtach łańcucha wejściowego w produkcie *SRCBUF* . Wartość musi być równa zero lub większa.

SRCBUF (łańcuch znakowy 1-bajtowy x SRCLen)-dane wejściowe

Łańcuch, który ma zostać przekształcony.

Jest to bufor zawierający łańcuch, który ma zostać przekształcony z jednego zestawu znaków na inny.

TGTCSI (10-cyfrowa liczba całkowita ze znakiem)-wejście

Identyfikator kodowanego zestawu znaków łańcucha po konwersji.

Jest to identyfikator kodowanego zestawu znaków zestawu znaków, do którego ma zostać przekształcona wartość *SRCBUF* .

TGTLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość buforu wyjściowego.

Jest to długość (w bajtach) bufora wyjściowego *TGTBUF* ; Wartość musi być równa zero lub większa. Wartość ta może być mniejsza lub większa niż *SRCLen*.

TGTBUF (1-bajtowy łańcuch znaków x TGTLEN)-wyjście

Łańcuch po konwersji.

Jest to łańcuch po przekształceniu go w zestaw znaków zdefiniowany przez produkt *TGTCSI*. Przekształcony łańcuch może być krótszy lub dłuższy niż łańcuch bez konwersji. Parametr *DATLEN* wskazuje liczbę zwróconych poprawnych bajtów.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Długość łańcucha wyjściowego.

Jest to długość łańcucha zwracanego w buforze wyjściowym *TGTBUF*. Przekształcony łańcuch może być krótszy lub dłuższy niż łańcuch bez konwersji.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Kod zakończenia.

Jest to jedna z poniższych nazw:

CCOK

Zakończenie powiodło się.

CCWARN

Ostrzeżenie (częściowe zakończenie).

CCFAIL

Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny kwalifikujący *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

RCBRAK

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CMPCOD* to CCWARN:

RC2120

(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

Jeśli *CMPCOD* to CCFAIL:

RC2010

(2010, X'7DA') Parametr długości danych nie jest poprawny.

RC2150

(2150, X'866 ') Łańcuch DBCS nie jest poprawny.

RC2018

(2018, X'7E2') Uchwyt połączenia nie jest poprawny.

RC2046

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

RC2102

(2102, X'836 ') Niewystarczająca ilość dostępnych zasobów systemowych.

RC2145

(2145, X'861 ') Parametr buforu źródłowego jest niepoprawny.

RC2111

(2111, X'83F') Identyfikator kodowanego zestawu znaków źródła nie jest poprawny.

RC2112

(2112, X'840 ') Nie rozpoznano kodowania liczb całkowitych źródła.

RC2143

(2143, X'85F') Parametr długości źródła nie jest poprawny.

RC2071

(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci masowej.

RC2146

(2146, X'862 ') Parametr buforu docelowego jest niepoprawny.

RC2115

(2115, X'843 ') Identyfikator kodowanego zestawu znaków docelowych nie jest poprawny.

RC2116

(2116, X'844 ') Docelowe kodowanie liczb całkowitych nie zostało rozpoznane.

RC2144

(2144, X'860 ') Parametr długości docelowej nie jest poprawny.

RC2195

(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [“Kody powrotu dla IBM i \(ILE RPG\)”](#) na stronie 1455.

Wywołanie RPG (ILE)

```
C*..1.....2.....3.....4.....5.....6.....7..  
C                                CALLP      MQXCNVC(HCONN : OPTS : SRCCSI :  
C                                SRCLen : SRCBUF : TGTCSE :  
C                                TGTLEN : TGTFUF : DATLEN :  
C                                CMPCOD : REASON)
```

Definicja prototypu dla wywołania to:

```
D*..1.....2.....3.....4.....5.....6.....7..  
DMQXCNVC      PR      EXTPROC('MQXCNVC')  
D* Connection handle  
D HCONN              10I 0 VALUE  
D* Options that control the action of MQXCNVC  
D OPTS              10I 0 VALUE  
D* Coded character set identifier of string before conversion  
D SRCCSI            10I 0 VALUE  
D* Length of string before conversion  
D SRCLen            10I 0 VALUE  
D* String to be converted  
D SRCBUF            *   VALUE  
D* Coded character set identifier of string after conversion  
D TGTCSE            10I 0 VALUE  
D* Length of output buffer  
D TGTLEN            10I 0 VALUE  
D* String after conversion  
D TGTFUF            *   VALUE  
D* Length of output string  
D DATLEN            10I 0  
D* Completion code  
D CMPCOD            10I 0  
D* Reason code qualifying CMPCOD  
D REASON            10I 0
```

MQCONVX-wyjście konwersji danych

Ta definicja wywołania opisuje parametry, które są przekazywane do wyjścia konwersji danych.

Menedżer kolejek nie zawiera punktu wejścia o nazwie MQCONVX (patrz uwaga o składni [“11”](#) na stronie 1484).

Ta definicja jest częścią interfejsu DCI (Data Conversion Interface) produktu IBM MQ, który jest jednym z interfejsów środowiska produktu IBM MQ.

- [“Składnia”](#) na stronie 1482
- [“Użycie notatek”](#) na stronie 1483
- [“Parametry”](#) na stronie 1484
- [“Wywołanie RPG \(ILE\)”](#) na stronie 1485

Składnia

MQCONVX (MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF)

Użycie notatek

1. Wyjście konwersji danych jest to wyjście pisane przez użytkownika, które odbiera sterowanie podczas przetwarzania wywołania MQGET. Funkcja wykonywana przez wyjście konwersji danych jest zdefiniowana przez dostawcę wyjścia, jednak wyjście musi być zgodne z regułami opisanymi w tym miejscu oraz w powiązanej strukturze parametrów MQDXP.

Języki programowania, które mogą być używane na potrzeby wyjścia konwersji danych, są określane przez środowisko.

2. Wyjście jest wywoływane tylko wtedy, gdy *wszystkie* z następujących elementów są prawdziwe:

- Opcja GMCONV jest określona w wywołaniu MQGET
- Pole *MDFMT* w deskrytorze komunikatu nie ma wartości FMNONE.
- Komunikat nie znajduje się już w wymaganej reprezentacji, to znaczy jeden lub oba komunikaty *MDCSI* i *MDENC* różnią się od wartości określonej przez aplikację w deskrytorze komunikatu dostarczonym w wywołaniu MQGET.
- Menedżer kolejek nie wykonał jeszcze pomyślnie konwersji
- Długość buforu aplikacji jest większa od zera
- Długość danych komunikatu jest większa od zera
- Do tej pory kod przyczyny w operacji MQGET ma wartość RCNONE lub RC2079 .

3. Po zapisaniu wyjścia należy rozważyć kodowanie wyjścia w sposób, który umożliwi jego przekształcenie komunikatów, które zostały obcięte. Obcięte komunikaty mogą pojawić się w następujący sposób:

- Aplikacja odbierający udostępnia bufor, który jest mniejszy niż komunikat, ale określa opcję GMATM w wywołaniu MQGET.

W takim przypadku pole *DXREA* w parametrze *MQDXP* na wejściu do wyjścia będzie miało wartość RC2079.

- Nadawca wiadomości obciął ją przed wysłaniem. Może się to zdarzyć w przypadku komunikatów raportu, na przykład (więcej informacji na ten temat zawiera sekcja [“Konwersja komunikatów raportu”](#) na stronie 1471).

W takim przypadku pole *DXREA* w parametrze *MQDXP* na wejściu do wyjścia będzie miało wartość RCNONE (jeśli aplikacja odbierający udostępniła bufor, który był na tyle duży, aby komunikat został wyświetlony).

Z tego powodu wartość pola *DXREA* na wejściu do wyjścia nie może być zawsze używana do określenia, czy komunikat został obcięty.

Cechą wyróżniającą obciętą wiadomość jest to, że długość podana do wyjścia w parametrze *INLEN* będzie *mniejsza niż* długość wynikająca z nazwy formatu zawartej w polu *MDFMT* w deskrytorze komunikatu. Wyjście powinno więc sprawdzić wartość *INLEN* przed próbą przekształcenia danych; wyjście *nie powinno* zakładać, że pełna ilość danych implikowanych przez nazwę formatu została podana.

Jeśli wyjście *nie* zostało zapisane w celu przekształcenia obciętych komunikatów, a wartość *INLEN* jest mniejsza niż oczekiwana, wyjście powinno zwrócić wartość XRFAIL w polu *DXRES* parametru *MQDXP* , w polu *DXCC* ustawionym na CCWARN, a w polu *DXREA* ustawionym na wartość RC2110.

Jeśli wyjście *ma* zostało zapisane w celu przekształcenia obciętych komunikatów, wyjście powinno zostać przekształcone w jak największą ilość danych (patrz uwaga na następne użycie), starając się nie próbować badać ani konwertować danych poza końcem programu *INBUF*. Jeśli konwersja zakończy się pomyślnie, wyjście powinno pozostawić pole *DXREA* w parametrze *MQDXP* bez zmian. Zwraca wartość RC2079 , jeśli komunikat został obcięty przez menedżera kolejek odbiornika, a wartość RCNONE, jeśli komunikat został obcięty przez nadawcę komunikatu.

Możliwe jest również, że komunikat może rozwinąć *podczas* konwersji do punktu, w którym jest on większy niż *OUTBUF*. W takim przypadku wyjście musi zdecydować, czy komunikat ma zostać obcięty, a pole *DXAOP* w parametrze *MQDXP* wskaże, czy aplikacja odbierający określiła opcję GMATM.

4. Ogólnie zaleca się, aby wszystkie dane w komunikacie dostarczonym do wyjścia w produkcie *INBUF* były przekształcane, lub że żaden z nich nie jest. Wyjątkiem od tego jest jednak, jeśli komunikat jest obcinany przed konwersją lub podczas konwersji; w tym przypadku może wystąpić niepełny element na końcu bufora (na przykład: jeden bajt znaku dwubajtowego lub 3 bajty 4-bajtowej liczby całkowitej). W takiej sytuacji zaleca się pominięcie niekompletnego elementu, a nieużywane bajty w programie *OUTBUF* mają wartość null. Jednak pełne elementy lub znaki w tablicy lub łańcuchu powinny być przekształcane.
5. Gdy wyjście jest wymagane po raz pierwszy, menedżer kolejek próbuje załadować obiekt o takiej samej nazwie, jak format (poza rozszerzeniami). Załadowany obiekt musi zawierać wyjście, które przetwarza komunikaty z tą nazwą formatu. Zaleca się, aby nazwa wyjścia i nazwa obiektu, które zawierają wyjście, były identyczne, chociaż nie wszystkie środowiska wymagają tego.
6. Nowa kopia wyjścia jest ładowana, gdy aplikacja próbuje pobrać pierwszy komunikat, który używa tego produktu *MDFMT* od momentu połączenia aplikacji z menedżerem kolejek. Nowa kopia może być również załadowana w innym czasie, jeśli menedżer kolejek odrzucił wcześniej załadowaną kopię. Z tego powodu wyjście nie powinno próbować używać statycznej pamięci masowej do przekazywania informacji z jednego wywołania wyjścia do następnego-wyjście może być rozładowane między dwoma wywołaniami.
7. Jeśli istnieje wyjście podane przez użytkownika o tej samej nazwie co jeden z wbudowanych formatów obsługiwanych przez menedżer kolejek, wyjście podane przez użytkownika nie zastępuje wbudowanej procedury konwersji. Jedynymi okolicznościami, w których takie wyjście jest wywołane, są:
 - Jeśli wbudowana procedura konwersji nie może obsłużyć konwersji do lub z *MDCSI* lub *MDENC* biorących udział, lub
 - Jeśli wbudowana procedura konwersji nie przekształci danych (na przykład, ponieważ istnieje pole lub znak, które nie mogą zostać przekształcone).
8. Zasięg wyjścia jest zależny od środowiska. Nazwy produktu *MDFMT* powinny być wybierane w taki sposób, aby zminimalizować ryzyko wystąpienia starć z innymi formatami. Zaleca się, aby rozpoczynały się od znaków, które identyfikują aplikację definiującą nazwę formatu.
9. Wyjście konwersji danych działa w środowisku takim jak program, który wywołał wywołanie *MQGET*; środowisko obejmuje przestrzeń adresową i profil użytkownika (jeśli ma to zastosowanie). Program może być agentem kanału komunikatów wysyłającym komunikaty do docelowego menedżera kolejek, który nie obsługuje konwersji komunikatów. Wyjście nie może naruszać integralności menedżera kolejek, ponieważ nie jest ono uruchamiane w środowisku menedżera kolejek.
10. Jedynym wywołaniem *MQI*, który może być używany przez wyjście, jest *MQXCNV*; próba użycia innych wywołań *MQI* kończy się niepowodzeniem z kodem przyczyny *RC2219* lub innymi nieprzewidywalnymi błędami.
11. Menedżer kolejek nie zawiera punktu wejścia o nazwie *MQCONVX*. Nazwa wyjścia powinna być taka sama, jak nazwa formatu (nazwa zawarta w polu *MDFMT* w strukturze *MQMD*), chociaż nie jest to wymagane we wszystkich środowiskach.

Parametry

Wywołanie *MQCONVX* ma następujące parametry:

MQDXP (MQDXP)-wejście/wyjście

Blok parametru wyjścia konwersji danych.

Struktura ta zawiera informacje związane z wywoływaniem wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać wynik konwersji. Szczegółowe informacje na temat pól w tej strukturze można znaleźć w sekcji [“MQDXP-Dane-parametr wyjścia konwersji danych” na stronie 1472](#).

MQMD (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Po wejściu do wyjścia jest to deskryptor komunikatu, który zostałby zwrócony do aplikacji, jeśli nie wykonano żadnej konwersji. W związku z tym zawiera on *MDFMT*, *MDENC* i *MDCSI* nieprzekształconego komunikatu zawartego w produkcie *INBUF*.

Uwaga: Parametr *MQMD* przekazany do wyjścia jest zawsze najnowszą wersją deskryptora *MQMD* obsługiwaną przez menedżer kolejek, który wywołuje wyjście. Jeśli wyjście ma być przenośne między różnymi środowiskami, wyjście powinno sprawdzić pole *MDVER* w programie *MQMD*, aby upewnić się, że pola, do których ma dostęp wyjście, znajdują się w strukturze.

W systemie IBM i wyjście jest przekazywane do programu *MQMD* w wersji version-2.

W przypadku wyjścia wyjście powinno zmienić pola *MDENC* i *MDCSI* na wartości żądane przez aplikację, jeśli konwersja powiodła się; zmiany te zostaną odzwierciedlone z powrotem do aplikacji. Wszelkie inne zmiany wprowadzone przez wyjście do struktury są ignorowane; nie są one odzwierciedlane z powrotem do aplikacji.

Jeśli wyjście zwraca wartość *XROK* w polu *DXRES* struktury *MQDXP*, ale nie powoduje zmiany pól *MDENC* lub *MDCSI* w deskrypcie komunikatu, menedżer kolejek zwraca dla tych pól wartości, których dane pola w strukturze *MQDXP* miały na wejściu do wyjścia.

INLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość (w bajtach) *INBUF*.

Jest to długość buforu wejściowego *INBUF* określa liczbę bajtów, które mają być przetwarzane przez wyjście. *INLEN* jest mniejszą od długości danych komunikatu przed konwersją, a także długość buforu udostępnionego przez aplikację w wywołaniu *MQGET*.

Wartość jest zawsze większa od zera.

INBUF (1-bajtowy łańcuch bitowy x INLEN)-wejście

Bufor zawierający nieprzekonwertowany komunikat.

Ten komunikat zawiera dane komunikatu przed konwersją. Jeśli wyjście nie jest w stanie przekształcić danych, menedżer kolejek zwraca zawartość tego buforu do aplikacji po zakończeniu wyjścia.

Uwaga: Wyjście nie powinno zmieniać *INBUF*; Jeśli ten parametr zostanie zmieniony, wyniki nie zostaną zdefiniowane.

OUTLEN (10-cyfrowa liczba całkowita ze znakiem)-wyjście

Długość (w bajtach) *OUTBUF*.

Jest to długość buforu wyjściowego *OUTBUF*, która jest taka sama, jak długość buforu udostępnianego przez aplikację w wywołaniu *MQGET*.

Wartość jest zawsze większa od zera.

OUTBUF (1-bajtowy łańcuch bitowy x OUTLEN)-wyjście

Bufor zawierający przekształcony komunikat.

W przypadku wyjścia z wyjścia, jeśli konwersja zakończyła się pomyślnie (zgodnie z wartością *XROK* w polu *DXRES* parametru *MQDXP*), program *OUTBUF* zawiera dane komunikatu, które mają zostać dostarczone do aplikacji, w żądanej reprezentacji. Jeśli konwersja nie powiodła się, wszystkie zmiany wprowadzone w tym buforze zostaną zignorowane.

Wywołanie RPG (ILE)

```
C*..1.....2.....3.....4.....5.....6.....7..
C                                CALLP    exitname(MQDXP : MQMD : INLEN :
C                                INBUF : OUTLEN : OUTBUF)
```

Definicja prototypu dla wywołania to:

```

D*.1.....2.....3.....4.....5.....6.....7..
Dexitname      PR      EXTPROC('exitname')
D* Data-conversion exit parameter block
D MQDXP              44A
D* Message descriptor
D MQMD              364A
D* Length in bytes of INBUF
D INLEN              10I 0 VALUE
D* Buffer containing the unconverted message
D INBUF              *    VALUE
D* Length in bytes of OUTBUF
D OUTLEN             10I 0 VALUE
D* Buffer containing the converted message
D OUTBUF             *    VALUE

```

Koniec interfejsu programistycznego wrażliwego na produkt

SOAP-odwołanie

Transport IBM MQ dla informacji o odwołaniu SOAP ułożonych alfabetycznie.

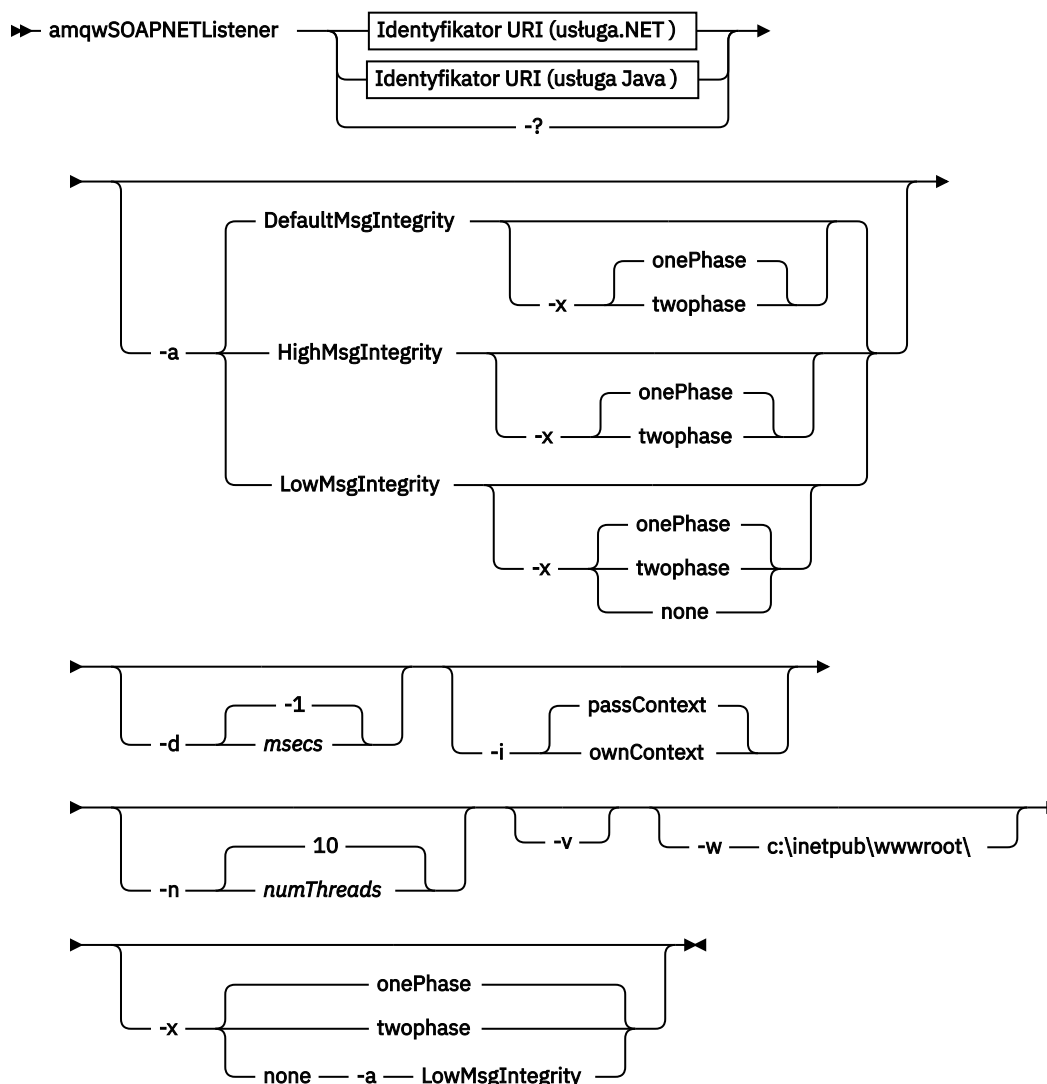
amqwSOAPNETListener: program nasłuchujący SOAP IBM MQ dla środowiska .NET 1 lub 2

Składnia i parametry dla programu nasłuchującego SOAP IBM MQ dla .NET Framework 1 lub 2.

Przeznaczenie

Uruchamia program nasłuchujący SOAP produktu IBM MQ dla środowiska .NET 1 lub 2.

.NET



Wymagane parametry

URI platforma

Patrz [“Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service” na stronie 1526.](#)

-?

Wydrukuj tekst pomocy opisujący sposób użycia komendy.

Parametry opcjonalne

-a integrityOption

Opcja *integrityOption* określa zachowanie obiektów nasłuchiwanie SOAP produktu IBM MQ, jeśli nie jest możliwe umieszczenie komunikatu żądania, który nie powiódł się w kolejce niedostarczonych komunikatów. *integrityOption* może przyjmować jedną z następujących wartości:

DefaultMsgIntegrity

W przypadku komunikatów nietrwałych program nasłuchujący wyświetla komunikat ostrzegawczy i kontynuuje wykonywanie, gdy oryginalny komunikat jest odrzucany. W przypadku komunikatów trwałych wyświetlany jest komunikat o błędzie, który powoduje utworzenie kopii zapasowej komunikatu żądania w taki sposób, aby pozostał on w kolejce żądań i kończy działanie. DefaultMsgIntegralność ma zastosowanie, jeśli pominięto opcję -a lub jeśli opcja *integrityOption* nie została określona.

LowMsgIntegrity

Zarówno dla komunikatów trwałych, jak i nietrwałych program nasłuchujący wyświetla ostrzeżenie i kontynuuje wykonywanie, odrzucając komunikat.

HighMsgIntegrity

W przypadku komunikatów trwałych i nietrwałych program nasłuchujący wyświetla komunikat o błędzie, tworzy kopię zapasową komunikatu żądania w taki sposób, że pozostaje w kolejce żądań i kończy działanie.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Jeśli zostanie podana wartość -x none , należy podać wartość -a LowMsgIntegrity . Jeśli flagi są niezgodne, program narzędziowy wdrażania zakończy działanie z komunikatem o błędzie i nie zostało wykonane żadne kroki wdrażania.

-d msec

msecs określa liczbę milisekund dla programu nasłuchującego SOAP produktu IBM MQ , który ma pozostać przy życiu, jeśli komunikaty żądania zostały odebrane w dowolnym wątku. Jeśli parametr *msecs* ma wartość -1, program nasłuchujący pozostanie aktywny na czas nieokreślony.

-i Kontekst

Kontekst określa, czy obiekty nasłuchiwanie przekazują kontekst tożsamości. *Kontekst* przyjmuje następujące wartości:

passContext

Ustaw kontekst tożsamości oryginalnego komunikatu żądania w komunikacie odpowiedzi. Program nasłuchujący SOAP sprawdza, czy ma uprawnienia do zapisywania kontekstu z kolejki żądań i przekazywania go do kolejki odpowiedzi. Podczas uruchamiania kolejki żądań w celu zapisania kontekstu oraz kolejki odpowiedzi do przekazywania kontekstu są wykonywane sprawdzanie w czasie wykonywania. Jeśli nie ma wymaganych uprawnień lub wywołanie MQOPEN nie powiedzie się, a komunikat odpowiedzi nie zostanie przetworzony. Komunikat odpowiedzi jest umieszczany w kolejce niedostarczonych komunikatów z nagłówkiem niedostarczonych komunikatów, który zawiera kod powrotu z uszkodzonego MQOPEN. Następnie program nasłuchujący kontynuuje przetwarzanie kolejnych komunikatów przychodzących w normalny sposób.

ownContext

Obiekt nasłuchiwanie SOAP nie przekazuje kontekstu. Zwrócony kontekst odzwierciedla ID użytkownika, pod którym nasłuchiwanie jest uruchomione, a nie identyfikator użytkownika, który utworzył oryginalny komunikat żądania.

Pola w kontekście pochodzenia są ustawiane przez menedżer kolejek, a nie przez obiekt nasłuchiwanie SOAP.

-n numThreads

numThreads określa liczbę wątków w wygenerowanym skrypcie startowym dla programu nasłuchującego SOAP produktu IBM MQ . Wartość domyślna wynosi 10. Jeśli przepustowość komunikatów jest wysoka, należy rozważyć zwiększenie tej liczby.

-v

-v ustawia szczegółowe dane wyjściowe z komend zewnętrznych. Komunikaty o błędach są zawsze wyświetlane. Komenda -v służy do wyprowadzania komend, które można dostosować do niestandardowych skryptów wdrażania.

-w serviceDirectory

serviceDirectory to katalog zawierający usługę Web Service.

-x transakcyjność

transakcyjność określa typ sterowania transakcyjnego dla obiektu nasłuchiwanie. *transakcyjność* można ustawić na jedną z następujących wartości:

onePhase

IBM MQ jest używana obsługa jednofazowa. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania zostanie ponownie dostarczony do aplikacji. Transakcje IBM MQ zapewniają, że komunikaty odpowiedzi są zapisywane dokładnie jeden raz.

twoPhase

Używana jest obsługa dwufazowa. Jeśli usługa jest odpowiednio napisana, komunikat jest dostarczany dokładnie jeden raz, koordynowany z innymi zasobami, w ramach pojedynczego zatwierdzonego wykonania usługi. Ta opcja ma zastosowanie tylko do połączeń z powiązaniami serwera.

none

Brak obsługi transakcyjnej. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania może zostać utracony, nawet jeśli jest on trwały. Możliwe, że usługa została wykonana lub nie została wykonana, a odpowiedź, raport lub komunikaty o niedostarczonych literach mogą, ale nie, być zapisywane.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a. Szczegółowe informacje można znaleźć w opisie opcji -a.

.NET Przykład

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

amqswsd1: generowanie pliku WSDL dla usługi .NET Framework 1 lub 2

Produkt **amqswsd1** przyjmuje usługę Web Service napisanej dla środowiska .NET Framework 1 lub 2 i generuje plik WSDL dla klasy, wstawiając identyfikator URI udostępnianego dla transportu produktu IBM MQ dla protokołu SOAP do wygenerowanego pliku WSDL.

Przeznaczenie

Produkt **amqswsd1** służy do generowania pliku WSDL zawierającego identyfikator URI usługi wdrożonej w produkcie IBM MQ. Za pomocą pliku WSDL można generować proxy klientów.

➡ amqswsd1 — *escapedUri* — *className* — .asmx — *className* — .wsdl ➡

Parametry

escapedUri (Wejście)

Identyfikator URI usługi, z którym wszystkie "&" zostały zmienione na "&". Na przykład:

```
"jms:/queue?destination=REQUESTDOTNET
&initialContextFactory=com.ibm.mq.jms.Nojndi
&connectionFactory=(connectQueueManager(QM1)binding(server))
&targetService=Quote.asmx"
```

className.asmx (dane wejściowe)

Klasa usługi.

className.wsdl (Wyjście)

Plik WSDL usługi.

Opis

Jeśli klasa jest implementowana przy użyciu modelu programowania za pomocą kodu, należy zbudować produkt *className.dll* i zapisać go w produkcie *./bin*.

amqwclientconfig: utwórz deskryptor wdrażania klienta usług Web Service Axis 1.4 dla transportu IBM MQ dla protokołu SOAP

Produkt **amqwclientconfig** tworzy plik deskryptora wdrażania klienta produktu `client-config.wsdd` Axis 1.4.

Przeznaczenie

Dodaje on transport `jms:/` do deskryptora i rejestruje `java:com.ibm.mq.soap.transport.jms.WMQSender` jako klasę do obsługi żądań SOAP dla transportu `jms:.`

Składnia

➤ `amqwclientconfig` ➤

Opis

amqwclientconfig wywołuje **amqwsetcp**, aby ustawić zmienną `CLASSPATH` i uruchamia komendę:

```
java org.apache.axis.utils.Admin client "%WMQSOAP_HOME%\bin\amqwclientTransport.wsdd"
```

amqwdeployWMQService: wdrażanie programu narzędziowego usług Web Service

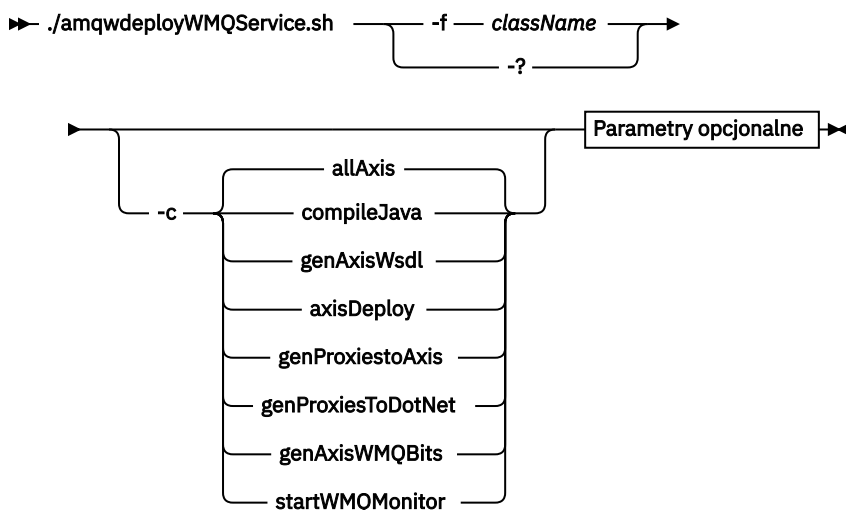
Narzędzie wdrażające przygotowuje klasę usługi do użycia jako usługa Web Service przy użyciu produktu IBM MQ jako transportu.

Przeznaczenie

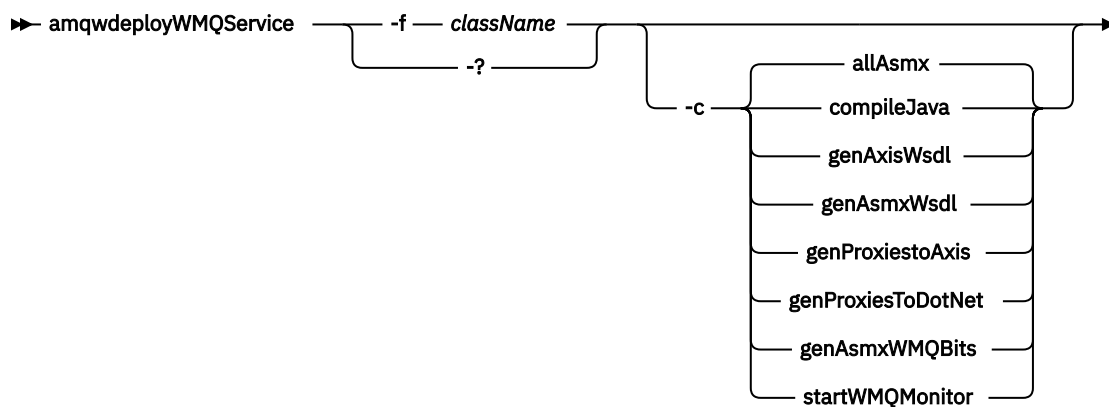
Użyj programu narzędziowego do wdrażania w celu wygenerowania plików potrzebnych do wdrożenia usługi środowiska Axis 1.4, .NET Framework 1 lub .NET Framework 2. Użyj tych plików, aby wdrożyć usługę wywoływaną przez produkt IBM MQ. Pliki wygenerowane przez program **amqwdeployWMQService** są wyświetlane w programie [“Pliki wyjściowe z produktu amqwdeployWMQService”](#) na stronie 1495.

Diagram składni

Systemy UNIX i Linux

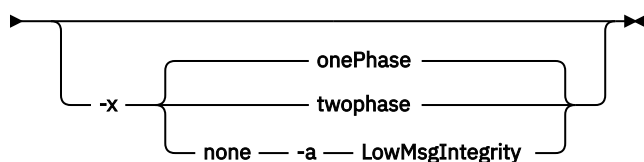
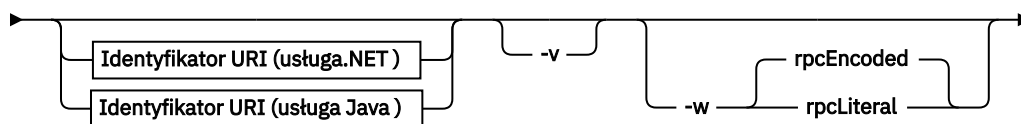
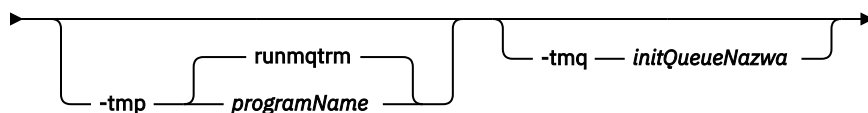
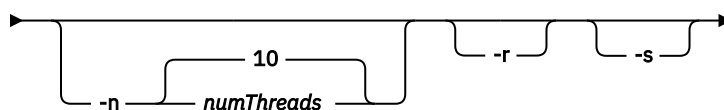
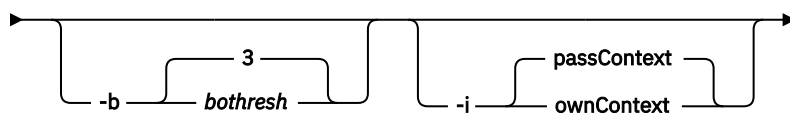
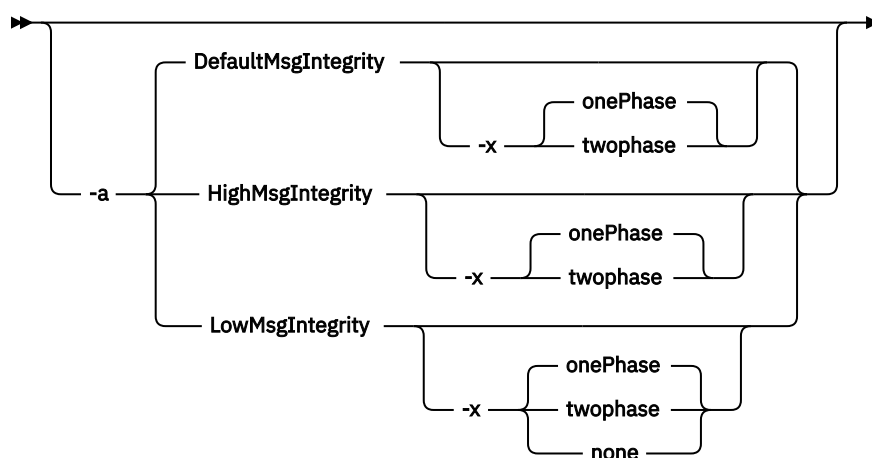


Windows



Parametry opcjonalne

Parametry opcjonalne



Wymagane parametry

-f *className*

className to nazwa klasy, która ma zostać wdrożona. W przypadku usług Axis *className* jest to plik źródłowy produktu Java , a w przypadku usług produktu .NET plik .asmx . Rysunek 10 na stronie 1492 przedstawia wdrożenie usługi Axis i Rysunek 11 na stronie 1492 usługi .NET .

```
amqwdployWmqService -f javaDemos/service/StockQuoteAxis.java
```

Rysunek 10. Przykładowe wdrożenie usługi Axis

```
amqwdployWmqService -f StockQuoteDotNet.asmx
```

Rysunek 11. Przykładowe wdrożenie usługi .NET

W przypadku bazy danych Javawartość *className* musi być pełną nazwą pakietu. Może być ona określona jako nazwa ścieżki z separatorami katalogów lub jako nazwa klasy z separatorami okresu. Wygenerowana klasa znajduje się w katalogu `./generated/client/remote/path name` . W przypadku usługi .NET , mimo że można określić katalog, wygenerowane serwery proxy produktu Java są zawsze umieszczane w produkcie `./generated/client/remote/dotNetService`.

Jeśli identyfikator URI zostanie określony za pomocą opcji -u , a w obrębie identyfikatora URI zostanie określona wartość *targetService*, program narzędziowy do wdrażania sprawdza obiekt *className*. Wartość *className* musi być zgodna z nazwą *targetService*. Jeśli klasa i usługa nie są zgodne, program narzędziowy do wdrażania wyświetli komunikat o błędzie i kończy działanie.

-?

Wydrukuj tekst pomocy opisujący sposób użycia komendy.

Parametry opcjonalne

-a *integrityOption*

Opcja *integrityOption* określa zachowanie obiektów nasłuchiwanie SOAP produktu IBM MQ , jeśli nie jest możliwe umieszczenie komunikatu żądania, który nie powiódł się w kolejce niedostarczonych komunikatów. *integrityOption* może przyjmować jedną z następujących wartości:

DefaultMsgIntegrity

W przypadku komunikatów nietrwałych program nasłuchujący wyświetla komunikat ostrzegawczy i kontynuuje wykonywanie, gdy oryginalny komunikat jest odrzucany. W przypadku komunikatów trwałych wyświetlany jest komunikat o błędzie, który powoduje utworzenie kopii zapasowej komunikatu żądania w taki sposób, aby pozostał on w kolejce żądań i kończy działanie.

DefaultMsgIntegralność ma zastosowanie, jeśli pominięto opcję -a lub jeśli opcja *integrityOption* nie została określona.

LowMsgIntegrity

Zarówno dla komunikatów trwałych, jak i nietrwałych program nasłuchujący wyświetla ostrzeżenie i kontynuuje wykonywanie, odrzucając komunikat.

HighMsgIntegrity

W przypadku komunikatów trwałych i nietrwałych program nasłuchujący wyświetla komunikat o błędzie, tworzy kopię zapasową komunikatu żądania w taki sposób, że pozostaje w kolejce żądań i kończy działanie.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Jeśli zostanie podana wartość -x none , należy podać wartość -a LowMsgIntegrity . Jeśli flagi są niezgodne, program narzędziowy wdrażania zakończy działanie z komunikatem o błędzie i nie zostało wykonane żadne kroki wdrażania.

-b *bothresh*

bothresh określa ustawienie progu wycofania dla kolejki żądań. Domyślną wartością jest 3.

-c operacja

operacja określa, która część procesu wdrażania ma zostać wykonana. *operacja* to jedna z następujących opcji:

allAxis

Wykonaj wszystkie kroki kompilacji i konfiguracji dla usługi Axis lub Java ⁵.

compileJava

Skompiluj usługę Java : .java do .class.

genAxisWsd1

Wygeneruj plik WSDL: .class do .wsdl.

axisDeploy

Przeprowadź wdrożenie pliku klasy: .wsdl do .wsdd, zastosuj .wsdd.

genProxiestoAxis

Generate proxies: .wsdl to .java and .class.

genAxisWMBits

Konfigurowanie kolejek produktu IBM MQ , programów nasłuchujących SOAP IBM MQ i wyzwalaczy dla usługi Axis.

allAsmx

Wykonaj wszystkie kroki konfiguracji dla usługi .NET⁶.

genAsmxWsd1

Wygeneruj plik WSDL: .asmx do .wsdl.

genProxiesToDotNet

Generate proxies: .wsdl to .java, .class, .cs and .vb.

genAsmxWMBits

Konfigurowanie kolejek produktu IBM MQ , programów nasłuchujących i wyzwalaczy SOAP IBM MQ

startWQMMonitor

Uruchom monitor wyzwalacza dla usług SOAP IBM MQ .

Uwaga: `runmqtrm` jest uruchamiany pod ID użytkownika mqm . Jeśli zabezpieczenia są problemem, należy upewnić się, że obiekty nasłuchiwanie są uruchamiane pod odpowiednimi identyfikatorami użytkowników.

-i Kontekst

Kontekst określa, czy obiekty nasłuchiwanie przekazują kontekst tożsamości. *Kontekst* przyjmuje następujące wartości:

passContext

Ustaw kontekst tożsamości oryginalnego komunikatu żądania w komunikacie odpowiedzi. Program nasłuchujący SOAP sprawdza, czy ma uprawnienia do zapisywania kontekstu z kolejki żądań i przekazywania go do kolejki odpowiedzi. Podczas uruchamiania kolejki żądań w celu zapisania kontekstu oraz kolejki odpowiedzi do przekazywania kontekstu są wykonywane sprawdzanie w czasie wykonywania. Jeśli nie ma wymaganych uprawnień lub wywołanie MQOPEN nie powiedzie się, a komunikat odpowiedzi nie zostanie przetworzony. Komunikat odpowiedzi jest umieszczany w kolejce niedostarczonych komunikatów z nagłówkiem niedostarczonych komunikatów, który zawiera kod powrotu z uszkodzonego MQOPEN. Następnie program nasłuchujący kontynuuje przetwarzanie kolejnych komunikatów przychodzących w normalny sposób.

ownContext

Obiekt nasłuchiwanie SOAP nie przekazuje kontekstu. Zwrócony kontekst odzwierciedla ID użytkownika, pod którym nasłuchiwanie jest uruchomione, a nie identyfikator użytkownika, który utworzył oryginalny komunikat żądania.

⁵ Wartość domyślna, jeśli właściwość *className* ma rozszerzenie .java

⁶ Wartość domyślna, jeśli właściwość *className* ma rozszerzenie .asmx.

Pola w kontekście pochodzenia są ustawiane przez menedżer kolejek, a nie przez obiekt nasłuchiwanie SOAP.

-n numThreads

numThreads określa liczbę wątków w wygenerowanym skrypcie startowym dla programu nasłuchującego SOAP produktu IBM MQ. Wartość domyślna wynosi 10. Jeśli przepustowość komunikatów jest wysoka, należy rozważyć zwiększenie tej liczby.

-r

-r określa, że wszystkie istniejące definicje kolejki monitora żądania lub wyzwalacza są zastępowane. Kolejki monitora wyzwalacza są zastępowane tylko wtedy, gdy określono również parametr *-tmq*. Kolejki są ponownie tworzone przy użyciu standardowych atrybutów domyślnych, a istniejące komunikaty w kolejkach są usuwane. Jeśli opcja *-r* nie jest używana, wszystkie istniejące definicje kolejek nie są zmieniane, a istniejące komunikaty nie są usuwane. Nie podając opcji *-r*, należy upewnić się, że wszystkie dostosowane atrybuty kolejki są zachowywane.

-s

Skonfiguruj funkcję nasłuchiwanie tak, aby była uruchamiana jako usługa IBM MQ. Jeśli obie opcje *-s* i *-tmq* zostaną określone, narzędzie wdrażające wyświetli komunikat o błędzie i kończy działanie.

-tmp programName

programName określa nazwę programu monitora wyzwalacza. Opcji *-tmp programName* należy używać w środowisku UNIX lub Linux jako alternatywy dla korzystania z produktu **runmqtrm**. Programy, które inicjuje, działają pod uprawnieniem mqm.

Na przykład:

```
amqwdployMQService -f javaDemos/service/StockQuoteAxis.java  
-tmq trigger.monitor.queue -tmp trigmon
```

-tmq queueName

queueName określa nazwę kolejki monitora wyzwalacza. Definicje procesów produktu IBM MQ są tworzone w celu skonfigurowania automatycznego wyzwalania obiektów nasłuchiwanie SOAP produktu IBM MQ przy użyciu powiązanej nazwy kolejki monitora wyzwalacza. Jeśli ta opcja nie zostanie podana, program narzędziowy wdrażający nie zdefiniuje konfiguracji wyzwalania. Jeśli obie opcje *-s* i *-tmq* zostaną określone, narzędzie wdrażające wyświetli komunikat o błędzie i kończy działanie.

URI platforma

Patrz [“Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service”](#) na stronie 1526.

-v

-v ustawia szczegółowe dane wyjściowe z komend zewnętrznych. Komunikaty o błędach są zawsze wyświetlane. Komenda *-v* służy do wyprowadzania komend, które można dostosować do niestandardowych skryptów wdrażania.

-w

-w służy do określania stylu pliku WSDL do wygenerowania. Wartość domyślna to *rpcEnclosed*, aby zapewnić kompatybilność z poprzednimi wersjami transportu produktu IBM MQ dla protokołu SOAP. Użyj komendy *rpcLiteral*, aby utworzyć plik WSDL zgodny z generowaniem proxy klienta Axis2. *rpcEncoded* nie jest kompatybilny z rekomendacjami WS-I.

-x transakcyjność

transakcyjność określa typ sterowania transakcyjnego dla obiektu nasłuchiwanie. *transakcyjność* można ustawić na jedną z następujących wartości:

onePhase

IBM MQ jest używana obsługa jednofazowa. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania zostanie ponownie dostarczony do aplikacji. Transakcje IBM MQ zapewniają, że komunikaty odpowiedzi są zapisywane dokładnie jeden raz.

twoPhase

Używana jest obsługa dwufazowa. Jeśli usługa jest odpowiednio napisana, komunikat jest dostarczany dokładnie jeden raz, koordynowany z innymi zasobami, w ramach pojedynczego

zatwierdzonego wykonania usługi. Ta opcja ma zastosowanie tylko do połączeń z powiązaniami serwera.

none

Brak obsługi transakcyjnej. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania może zostać utracony, nawet jeśli jest on trwały. Możliwe, że usługa została wykonana lub nie została wykonana, a odpowiedź, raport lub komunikaty o niedostarczonych literach mogą, ale nie, być zapisywane.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Szczegółowe informacje można znaleźć w opisie opcji -a .

Błędy

Jeśli w systemie Windowszgłaszane są błędy związane z produktem **amqswsdl**, spróbuj wydać następującą komendę, aby zarejestrować pliki produktu .asmx jako usługi.

```
%windir%/Microsoft.NET/Framework/ version number /aspnet_regiis.exe -ir
```

Problem występuje zwykle w systemach, w których nie zainstalowano serwera IIS, lub gdy serwer IIS został zainstalowany po NET. Problem ten występuje, gdy program **amqswsdl** generuje pliki .wsdl .

Uwaga: Klucze rejestru są również wymagane, aby umożliwić nastuchiwanie wywoływanie usług. Jeśli używane są własne, niestandardowe procedury wdrażania, problem może nie zostać napotkany do czasu uruchomienia.

Pliki wyjściowe z produktu amqwdeployWMQService

Lista danych wyjściowych katalogów i plików z programu **amqwdeployWMQService**

Tabela 183. Pliki wyjściowe z produktu amqwdeployWMQService			
Dane wyjściowe	Opis	Katalog wyjściowy	Nazwa pliku
.class	Skompilowany plik źródłowy Java	./generated/server/server package	classname.class
.wsdl	opis usługi	./generated	className Axis_Wmq.wsdl className DotNet_Wmq.wsdl
.wsdd	Pliki wdrożenia klienta i usługi Axis	./	client-config.wsdd server-config.wsdd
		./generated/server/server package	className _deploy.wsdd className _undeploy.wsdd

Tabela 183. Pliki wyjściowe z produktu amqwdeployWMQService (kontynuacja)			
Dane wyjściowe	Opis	Katalog wyjściowy	Nazwa pliku
Źródło klienta (.vb, .cs, .java)	Kody pośredniczące klienta .Net do usługi Axis	./generated/client	classname AxisService.cs classname AxisService.vb
	Kody pośredniczące klienta .Net do .Net	./generated/client	classname DotNet.cs classname DotNet.vb
Program pomocy klienta (.java i .class)	Serwery proxy klienta Java do usługi sieciowej .Net	./generated/server/soap/client/remote/dotnetService	className DotNet.class className DotNet.java className DotNetLocator.class className DotNetLocator.java className DotNetSoap12Stub.class className DotNetSoap12Stub.java className DotNetSoap_BindingStub.class className DotNetSoap_BindingStub.java className DotNetSoap_PortType.class className DotNetSoap_PortType.java
	Serwery proxy klienta produktu Java do usługi Axis	./generated/server/soap/client/remote/client package	SoapServer className AxisBindingSoapStub.class SoapServer className AxisBindingSoapStub.java className Axis.class className Axis.java className AxisService.class className AxisService.java className AxisServiceLocator.class className AxisServiceLocator.java
Skrypty (.cmd i .sh)	Skrypty nastęchiwania	/generated/server	startWMQJListener.cmd startWMQJListener.sh startWMQNListener.cmd endWMQJListener.cmd endWMQJListener.sh endWMQNListener.cmd

Uwagi dotyczące używania produktu amqwdeployWMQService

Opisuje zadania wykonywane przez produkt **amqwdeployWMQService**.

Program narzędziowy do wdrażania wykonuje następujące działania.

1. Sprawdza ścieżki do następujących plików:

- axis.jar.
- WMQSOAP_HOME/java/lib/com.ibm.mq.soap.jar.

- W systemie Windows, `csc.exe`
2. W systemie Windows używa się produktu `%SystemRoot%\Microsoft.NET\Framework\v1.1.432` lub, jeśli jest zainstalowany kompilator C#, ścieżki do produktu `csc.exe` jako ścieżki do środowiska produktu .NET.

Uwaga: Jeśli zainstalowany jest produkt Microsoft Visual Studio 2008 (wersja 9), produkt `wsdl.exe` nie znajduje się w ścieżce do produktu `csc.exe`. Należy dodać ścieżkę do zmiennej ścieżki .NET do zmiennej ścieżki, na przykład:

```
Set Path=C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;%Path%
```

3. Tworzy katalog `./generated` i wymagane podkatalogi, jeśli nie istnieją.
4. W przypadku usług Java kompiluje źródło w klasie `className.class`.
5. Generuje plik WSDL.
6. W przypadku usług produktu Java tworzy pliki deskryptora wdrażania `className_deploy.wsdd` i `className_undeploy.wsdd`.
7. W przypadku usług produktu Java tworzy lub aktualizuje plik deskryptora wdrażania produktu Axis (`server-config.wsdd`).
8. Generuje proxy klienta dla produktów Java, C# i Visual Basic z pliku WSDL.

Uwaga: W systemie Windows program narzędziowy do wdrażania generuje proxy dla Visual Basic i C# niezależnie od języka, w którym usługa jest napisana. Plik WSDL i wygenerowane proxy z niego zawierają odpowiedni identyfikator URI do wywołania usługi:

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
```

Rysunek 12. Przykładowy identyfikator URI w wygenerowanym kliencie .NET do wywołania usługi .NET

b.

```
jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
&connectionFactory=(connectQueueManager(WMQSOAP.DEMO.QM))
&initialContextFactory=com.ibm.mq.jms.NoJndi
&targetService=soap.server.StockQuoteAxis.java
&replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
```

Rysunek 13. Przykładowy identyfikator URI w wygenerowanym kliencie produktu .NET do wywołania usługi Axis 1

9. Kompiluje serwery proxy produktu Java.
10. Tworzy kolejkę IBM MQ, `requestQueue` do przechowywania żądań dla usługi. Domyślna nazwa kolejki ma postać `SOAPJ.directory` lub można podać wartość `requestQueue` w opcji -u URI.
11. Tworzy pliki skryptowe komend i powłoki w celu uruchomienia programów nasłuchujących SOAP programu IBM MQ, które przetwarzają kolejkę żądań.
12. Jeśli użyto opcji -tmq, program narzędziowy do wdrażania tworzy definicje IBM MQ w celu automatycznego wyzwolenia procesów nasłuchiwanie SOAP produktu IBM MQ.
 - Program narzędziowy do wdrażania korzysta z atrybutu `APPLICID` komendy `runmqsc DEFINE PROCESS` w celu określenia komendy uruchamiającego proces nasłuchujący. Komenda ta ma nazwę osadzonego w nim katalogu wdrażania. Pole `APPLICID` ma maksymalną długość wynoszącą 256, co ogranicza maksymalną długość katalogu wdrażania. Limit katalogu dla usług produktu Java jest następujący:

- Systemy UNIX i Linux : 218
- Windows: 197 minus długość nazwy kolejki żądań.

W przypadku usług .NET limit katalogu jest następujący:

- Windows: 209 minus długość nazwy usługi pomniejszona o rozszerzenie .asmx .
- Program narzędziowy do wdrażania sprawdza, czy limit dla APPLICID został przekroczony. Jeśli limit zostanie przekroczony, program narzędziowy nie podejmie próby zdefiniowania procesu wyzwalania. Zostanie wyświetlony komunikat o błędzie, a proces wdrażania nie powiedzie się bez wykonywania żadnych kroków wdrażania.

W poniższych przykładach przedstawiono komendy konfiguracji i uruchamiania wygenerowane przez program narzędziowy do wdrażania w celu uruchomienia programu nasłuchującego SOAP produktu IBM MQ .

```
DEFINE PROCESS(requestQueue) APPLICID(applicIDStr) REPLACE
ALTER QLOCAL (requestQueue) TRIGTYPE(FIRST) TRIGGER
PROCESS(requestQueue) INITQ(initQueueName) TRIGMPRI(0)
```

Rysunek 14. Komendy konfiguracyjne produktu IBM MQ służące do wyzwalania nasłuchiwanie SOAP.

```
applicIDStr = start "Java WMQSoapListener -requestQueue"
/min .\generated\server\startWMQJListener.cmd;
```

Rysunek 15. Uruchamianie programu nasłuchującego SOAP Axis w systemie Windows

```
applicIDStr = start "WMQAsmxListener -className\
/min .\generated\server\startWMQNListener.cmd;
```

Rysunek 16. Uruchamianie programu nasłuchującego SOAP produktu .NET w systemie Windows

```
applicIDStr = xterm -iconic -T "Java WMQSoapListener_requestQueue\"
-e ./generated/server/startWMQJListener.sh & #
```

Rysunek 17. Uruchamianie programu nasłuchującego SOAP Axis w systemach UNIX i Linux

amqwRegisterdotNet: zarejestruj transport IBM MQ dla protokołu SOAP do .NET

Zarejestruj produkt IBM MQ transport for SOAP do globalnej pamięci podręcznej zespołu na serwerze .NET.

Przeznaczenie

Produkt **amqwRegisterdotNet** rejestruje nadawcę SOAP IBM MQ , obiekt nasłuchiwanie SOAP i procesor WSDL przy użyciu środowiska .NET Framework 1 lub 2.

Składnia

➡ amqwRegisterdotNet ➡

Opis

Produkt **amqwRegisterdotNet** jest uruchamiany automatycznie podczas instalacji. Nie ma potrzeby ponownego uruchamiania go, jeśli używane środowisko .NET zostało zainstalowane przed transportem produktu IBM MQ dla protokołu SOAP. Możesz go uruchamiać tyle razy ile chcesz. Należy go użyć

do ponownego zarejestrowania transportu produktu IBM MQ dla protokołu SOAP z różnymi wersjami środowiska produktu .NET .

Uwaga: Na serwerze Windows 2003 należy także uruchomić program narzędziowy **aspnet_regiis** , nawet jeśli nie jest on wdrażany na serwerze Information Server (IIS). Położenie programu narzędziowego aspnet_regiis.exe może różnić się w zależności od wersji środowiska Microsoft.NET Framework, ale zwykle znajduje się on w: %SystemRoot%/Microsoft.NET/Framework/version number/aspnet_regiis. Jeśli zainstalowanych jest wiele wersji, należy użyć produktu **aspnet_regiis** dla używanej wersji środowiska .NET .

Licencja na oprogramowanie Apache

Apache License, wersja 2.0, styczeń 2004 <http://www.apache.org/licenses/>

<https://www.apache.org/licenses/>

Licencja Apache

Wersja 2.0, styczeń 2004

<http://www.apache.org/licenses/>

WARUNKI UŻYWANIA, ROZMNAŻANIA I DYSTRYBUCJI

1. Definicje.

"Licencja" oznacza warunki używania, reprodukcji, i rozprowadzanie zgodnie z sekcjami 1 do 9 niniejszego dokumentu.

"Licencjodawca" oznacza właściciela praw autorskich lub podmiot uprawniony przez właściciela praw autorskich, który udziela licencji.

"podmiot prawny" oznacza związek jednostki działającej i wszystkie inne podmioty, które kontrolują, są kontrolowane przez lub są pod wspólną kontrolą element sterujący z tym obiektem. Do celów niniejszej definicji,

"kontrola" oznacza (i) moc, bezpośrednią lub pośrednią, w celu spowodowania kierunku lub zarządzanie takim podmiotem, czy to na podstawie umowy, czy w przeciwnym wypadku lub (ii) własność 50% (50%) lub więcej wybitne udziały, lub (iii) korzystne prawo własności tego podmiotu.

"Ty" (lub "Twój") oznacza osobę prawną lub prawną wykonywania uprawnień przyznanych na mocy niniejszej Licencji.

"Formularz źródłowy" oznacza preferowaną formę dokonywania modyfikacji, uwzględnianie, ale nie ograniczanie się do kodu źródłowego oprogramowania, dokumentacji pliki źródłowe i konfiguracyjne.

"Obiekt" oznacza każdą formę wynikającą z mechanicznego przekształcenie lub tłumaczenie formularza źródłowego, w tym nie ogranicza się do skompilowanego kodu wynikowego, wygenerowanej dokumentacji, i przekształceń do innych typów nośników.

"Praca" oznacza pracę autorstwa, niezależnie od tego, czy jest ona w źródle, czy Formularz obiektu, udostępniony na podstawie Licencji, wskazany przez powiadomienia o prawach autorskich, które są dołączone do utworu lub dołączane do niego (przykład znajduje się w dodatku).

"Prace pochodne" oznaczają wszelkie prace, zarówno w źródle, jak i w obiekcie formularz, który jest oparty na (lub pochodnie) pracy i dla którego Korekty redakcyjne, adnotacje, opracowania lub inne modyfikacje reprezentują, jako całość, oryginalną pracę autorstwa. Do celów

z niniejszej Licencji, Zakłady Pochodne nie obejmują prac, które pozostają oddzielenie od lub jedynie powiązanie (lub powiązanie według nazwy) z interfejsami, Prace i Pochodne Prace z.

"Wkład" oznacza wszelkie prace autorskie, w tym oryginalna wersja Pracy oraz wszelkie modyfikacje lub uzupełnienia do tej roboty lub jej prace pochodne, to jest celowo wprowadzone do licencji na włączenie do pracy przez właściciela praw autorskich lub przez osobę lub osobę prawną upoważnioną do składania w imieniu właściciela praw autorskich. Do celów niniejszej definicji "przedłożony" oznacza dowolną formę komunikacji elektronicznej, werbalnej lub pisemnej na rzecz Licencjodawcy lub jej przedstawicieli, w tym, ale nie ograniczoną do komunikacji na elektronicznych listach mailingowych, systemy kontroli kodu źródłowego, i wydawać systemy śledzenia, które są zarządzane przez lub w imieniu Licencjodawcy w celu omówienia i usprawnienia pracy, ale wykluczanie komunikacji, która jest oznakowana w sposób wyraźny lub w inny sposób wyznaczone na piśmie przez właściciela praw autorskich jako "Nie jest to wkład".

"Kontrybutor" oznacza Licencjodawcę oraz każdą osobę lub podmiot prawny w imieniu którego wkład został otrzymany przez Licencjodawcę oraz następnie włączone do pracy.

2. Nadanie licencji na prawa autorskie. Z zastrzeżeniem warunków określonych w art. Niniejsza Licencja, każdy Kontrybutor niniejszym udziela Użytkownikom wieczystego, na całym świecie, niewyłączny, bez opłat, royalty-free, nieodwołalne licencja na prawa autorskie do powielania, przygotowania Pracowni Robót Budowlanych, publicznie wyświetlać, publicznie wykonuj, sublicencji i dystrybuuj Prace i takie prace pochodne w formie źródłowej lub obiektowej.

3. Nadanie licencji Patent. Z zastrzeżeniem warunków określonych w art. Niniejsza Licencja, każdy Kontrybutor niniejszym udziela Użytkownikom wieczystego, na całym świecie, niewyłączny, bez opłat, royalty-free, nieodwołalne (z wyjątkiem podanych w niniejszej sekcji) licencji patentowej do dokonania, wykonane, korzystać, oferować do sprzedaży, sprzedaży, importu, a w inny sposób przekazać pracę, w przypadku gdy licencja ta ma zastosowanie tylko do tych roszczeń patentowych licencjodawców przez takiego Kontrybutora, które są bezwzględnie naruszane przez ich Wkład (-y) samodzielnie lub w połączeniu z ich wkładem (-ami) z pracą, do której złożono takie składki. Jeśli instytutowe spory sądowe przeciwko jakiegokolwiek jednostce (w tym cross-claim lub counterclaim in a lawsuit) zarzucających, że praca lub wkład zawarty w ramach prac stanowi bezpośrednie lub przyczyniano się do naruszenia patentów, a następnie wszelkich licencji patentowych udzielonego Państwu na mocy niniejszej Licencji na tę pracę kończy się od dnia złożenia takiego sporu sądowego.

4. Redystrybucja. Użytkownik ma prawo kopiować i dystrybuować kopie Prace lub jego prace pochodne w dowolnym medium, z lub bez modyfikacji oraz w formularzu Source lub Object, pod warunkiem, że spełniają następujące warunki:

(a) Musisz dać innym odbiorcom pracy lub Instrument pochodny stanowi kopię niniejszej Licencji; oraz

(b) Musisz spowodować jakiegokolwiek zmodyfikowane pliki w celu noszenia prominentnych ogłoszeń stwierdzające, że pliki zostały zmienione; oraz

(c) Należy zachować, w formie źródłowej wszelkich Zakładów Pochodnych które rozprowadzasz, wszystkie prawa autorskie, patenty, znaki towarowe, i

powiadomienia o przypisaniu ze źródła w pracy,
wykluczanie tych powiadomień, które nie dotyczą żadnej części
Zakładów Pochodnych; oraz

(d) Jeśli Praca zawiera plik tekstowy "NOTICE" jako część jego dystrybucja, a następnie wszelkie prace pochodne, które rozprowadzasz muszą zawierać czytelną kopię zamieszczonych ogłoszeń o przypisach w takim pliku NOTICE, z wykluczeniem tych powiadomień, które nie dotyczą jakiegokolwiek części Zakładów Pochodnych, w co najmniej jednym w następujących miejscach: w pliku tekstowym NOTICE rozproszony w ramach Zakładów Pochodnych; w ramach formularza źródłowego lub dokumentację, jeżeli jest ona dostarczona wraz z Zakładami Pochodnymi; lub w obrębie wyświetlacza generowanego przez Zakłady Pochodne, jeżeli i wszędzie tam, gdzie zwykle pojawiają się takie powiadomienia. Spis treści Plik NOTICE ma wyłącznie charakter informacyjny i Nie należy modyfikować licencji. Możesz dodać swój własny wkład uwagi w ramach Zakładów Pochodnych, które są dystrybuowane, obok lub jako dodatek do tekstu OGŁOSZENIA z pracy, pod warunkiem że takie dodatkowe uwagi dotyczące wkładu nie mogą być interpretowane podczas modyfikowania licencji.

Możesz dodać swoje własne autorskie oświadczenie do swoich modyfikacji i mogą udostępniać dodatkowe lub różne warunki licencji do stosowania, reprodukcji lub dystrybucji modyfikacji Użytkownika, lub dla każdego takiego Zakładu Pochodnych jako całości, pod warunkiem, że Twoje użytkowanie, reprodukcja, a dystrybucja Praca w inny sposób jest zgodna z warunki określone w niniejszej Licencji.

5. Składanie wkładów. O ile Użytkownik nie oświadczy wyraźnie inaczej, każdy Wkład umyślnie złożony w celu włączenia do pracy przez Użytkownika Licencjodawca jest na warunkach i warunkach niniejszej Licencji, bez jakichkolwiek dodatkowych warunków. Bez względu na powyższe, nic w niniejszym dokumencie nie zastępuje ani nie zmienia. Warunki jakiegokolwiek odrębnej umowy licencyjnej, którą można było wykonać z Licencjodawcą w odniesieniu do takich składek.

6. Znaki towarowe. Niniejsza Licencja nie udziela pozwolenia na korzystanie z handlu nazwy, znaki towarowe, znaki usługowe lub nazwy produktów Licencjobiorcy, z wyjątkiem, gdy jest to wymagane dla rozsądnego i zwyczajowego stosowania w opisywaniu Początek pracy i ponowne generowanie zawartości pliku NOTICE.

7. Zastrzeżenie dotyczące gwarancji. O ile nie wymaga tego obowiązujące prawo lub zgodził się na piśmie, Licencjodawca zapewnia pracę (i każdy Kontrybutor udostępnia swoje elementy wnoszone) w BASIS "AS IS", BEZ GWARANCJI LUB WARUNKÓW JAKICHKOLWIEK GWARANCJI, wyraźnych lub Domniemane, w tym, bez ograniczeń, wszelkie gwarancje lub warunki NIENARUSZANIA, PRZYDATNOŚCI DO SPOŻYCIA, PRZYDATNOŚCI DO OKREŚLONEGO CELU LUB GWARANCJI, ŻE PUBLIKACJA TA NIE Określonych celów. Użytkownik ponosi wyłączną odpowiedzialność za określenie stosowność stosowania lub redystrybucji pracy i zakładania wszelkich Ryzyko związane z korzystaniem z uprawnień wynikających z niniejszej Licencji.

8. Ograniczenie odpowiedzialności. W żadnym wypadku i w żadnej teorii prawnej, czy w torcie (w tym zaniedbania), umowie, czy w inny sposób, o ile nie jest to wymagane przez obowiązujące prawo (takie jak celowe i rażąco nieumyślne akty prawne) lub uzgodnione na piśmie, każdy Kontrybutor będzie odpowiedzialność wobec Użytkownika za szkody, w tym wszelkie bezpośrednie, pośrednie, specjalne, przypadkowe, lub następne szkody o dowolnym charakterze, powstałe jako

wynik niniejszej Licencji lub brak możliwości wykorzystania lub niemożności korzystania z Pracy (w tym, ale nie ograniczając się do odszkodowania z tytułu utraty dobrej woli, stopka robocza, awaria komputera lub nieprawidłowego działania, lub dowolna innych szkód handlowych lub strat), nawet jeśli taki Kontrybutor został poinformowany o możliwości wystąpienia takich szkód.

9. Akceptowanie gwarancji lub dodatkowej odpowiedzialności. Podczas redystrybucji Praca lub Pochodne Prace z nich, Możesz wybrać do zaoferowania, i pobiera opłatę za, akceptację wsparcia, rękojmi, bezkarność, lub inne zobowiązania z tytułu odpowiedzialności i/lub prawa z tym związane Licencja. Jednakże, przyjmując takie zobowiązania, Użytkownik może działać tylko we własnym imieniu i na swoją wyłączną odpowiedzialność, nie w imieniu jakiegokolwiek innego Kontrybutora, i tylko jeśli zgadzasz się na indemnifikację, bronić, i trzymać każdego Kontrybutor nieszkodliwy dla każdej odpowiedzialności. Taki kontrybutor został poniesiony przez tego kontrybutora lub roszczenia z niego związane. Twojego przyjęcia jakiegokolwiek takiej gwarancji lub dodatkowej odpowiedzialności.

KONIEC TERMS I WARUNKI

DODATEK: W jaki sposób zastosować licencję Apache do pracy użytkownika.

Aby zastosować licencję Apache do swojej pracy, należy dołączyć następujące informacje: informacja o tabliczce, z polami ujętych w nawiasy kwadratowe "[]" zastąpiono własnymi informacjami identyfikacyjnymi. (Nie uwzględniaj nawiasy kwadratowe!) Tekst powinien być ujęty w odpowiednim miejscu składnia komentarza dla formatu pliku. Polecamy również Nazwa pliku lub klasy oraz opis przeznaczenia zostaną włączone do ta sama "drukowana strona" jak uwagi dotyczące praw autorskich do łatwiejszego identyfikację w archiwach innych firm.

Copyright [yyyy] [name of copyright owner]

Licencjonowany w ramach licencji Apache , wersja 2.0 ("Licencja");
Nie możesz używać tego pliku z wyjątkiem zgodności z Licencją.
Użytkownik może uzyskać kopię Licencji pod adresem

<http://www.apache.org/licenses/LICENSE-2.0>

O ile nie jest to wymagane przez obowiązujące przepisy prawa lub uzgodnione w formie pisemnej, oprogramowanie dystrybuowane na podstawie Licencji dystrybuowane są na "AS IS" BASIS, BEZ GWARANCJI LUB WARUNKÓW JAKICHKOLWIEK, wyraźnych czy domniemanych. Zapoznaj się z licencją dla konkretnego języka, któremu podlega uprawnienia, oraz ograniczeń wynikających z Licencji.

Ustawienia SOAP produktu MQMD

Nadawca SOAP IBM MQ i program nasłuchujący SOAP IBM MQ tworzą deskryptor komunikatu (**MQMD**). W tym temacie opisano pola, które należy ustawić w strukturze MQMD, jeśli tworzony jest własny nadawca lub program nasłuchujący SOAP.

Przeznaczenie

Wartości ustawione w produkcie **MQMD** sterują wymianą komunikatów między nadawcą SOAP IBM MQ , programem nasłuchującym SOAP IBM MQ i programem klienckim SOAP. Jeśli tworzony jest własny nadawca lub program nasłuchujący SOAP, należy postępować zgodnie z regułami w sekcji [Tabela 184 na stronie 1503](#).

Opis

W produkcie Tabela 184 na stronie 1503 opisano, w jaki sposób pola **MQMD** są ustawiane przez nadawcę SOAP IBM MQ i program nasłuchujący SOAP IBM MQ. Jeśli napiszesz własny nadawca lub program nasłuchujący, musisz ustawić te pola zgodnie z regułami wymiany komunikatów. Program nasłuchujący SOAP IBM MQ jest zgodny z typowymi protokołami wymiany komunikatów produktu IBM MQ. W przypadku pisania własnego nadawcy w celu pracy z obiektami nasłuchiwanymi SOAP IBM MQ można ustawić różne wartości **MQMD**.

W programie Tabela 184 na stronie 1503 wartości w kolumnie Ustawienia są zorganizowane w następujący sposób:

Żądanie, jednokierunkowe

Ustawienia wprowadzone przez nadawcę SOAP IBM MQ.

Odpowiedź, Raport

Ustawienia wprowadzone przez program nasłuchujący SOAP produktu IBM MQ w odpowiedzi na żądanie nadawcy SOAP IBM MQ.

ALL

Ustawienia wprowadzone zarówno przez nadajnik SOAP IBM MQ, jak i program nasłuchujący SOAP IBM MQ.

Niestandardowy nadawca

Możesz napisać własny nadawca. Zwykle niestandardowy nadawca przestania standardowe opcje raportu.

Tabela 184. Ustawienia SOAP MQMD		
Nazwa pola	Ustawienie	Wartości
<i>StrucId</i>	ALL MQMD_STRUC_ID	'MD-1' 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 Niestandardowy nadawca Patrz sekcja " Niestandardowe opcje raportu " na stronie 1507	52428800
<i>MsgType</i>	Żądanie MQMT_REQUEST Działanie MQMT_REPLY Raport MQMT_REPORT Jednokierunkowe MQMT_DATAGRAM	MQMT_REQUEST 1 MQMT_REPLY 2 MQMT_REPORT 4 MQMT_DATAGRAM 8

Tabela 184. Ustawienia SOAP MQMD (kontynuacja)

Nazwa pola	Ustawienie	Wartości
<i>Expiry</i>	Żądanie, jednokierunkowe Wartość określona przez opcję Expiry w identyfikatorze URI. Wartością domyślną jest MQEI_UNLIMITED. Działanie Wartość Expiry w komunikacie żądania Raport MQEI_UNLIMITED	MQEI_UNLIMITED -1
<i>Feedback</i>	Żądanie, odpowiedź, jednokierunkowe MQFB_NONE. Raport - Generowany przez menedżera kolejek- wartość ustawiona zgodnie ze zwykłym regułami. - Wygenerowane przez program nasłuchujący SOAP IBM MQ : MQRC_BACKOUT_THRESHOLD_REACHED Przekroczono próg wycofania dla wielu prób. MQRCCF_MD_FORMAT_ERROR Komunikat nie został rozpoznany jako posiadający nagłówek MQRFH2 . MQRC_RFH_PARM_MISSING Brak wymaganego parametru, na przykład SoapAction, w produkcie MQRFH2 . MQRC_RFH_FORMAT_ERROR Sprawdzenie integralności partycji MQRFH2 nie powiodło się, na przykład uszkodzenia wewnętrzne są uszkodzone. MQRC_RFH_ERROR Program MQRFH2 przeszedł kontrolę integralności, ale treść komunikatu nie jest ustawiona na wartość 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	Zależy od środowiska
<i>CodedCharSetId</i>	ALL Ustaw na UTF-8	1208

Tabela 184. Ustawienia SOAP MQMD (kontynuacja)

Nazwa pola	Ustawienie	Wartości
<i>Format</i>	Żądanie, odpowiedź, jednokierunkowe MQFMT_RF_HEADER_2 Raport Raporty menedżera kolejek Następujące reguły IBM MQ Raporty programu nasłuchującego SOAP produktu IBM MQ Format oryginalnego komunikatu żądania.	MQFMT_RF_HEADER_2 "MQRFH2 "
<i>Priority</i>	Żądanie, jednokierunkowe Określony przez opcję Priorytet w identyfikatorze URI. Wartością domyślną jest MQPRI_PRIORITY_AS_Q_DEF. Odpowiedź, Raport Wartość Priorytet w komunikacie żądania.	MQPRI_PRIORITY_AS_Q_DEF -1
<i>Persistence</i>	Żądanie, jednokierunkowe MQPER_PERSISTENCE_AS_Q_DEF. Odpowiedź, Raport Wartość Persistence w komunikacie żądania.	MQPER_PERSISTENCE_AS_Q_DEF 2
<i>MsgId</i>	Żądanie, jednokierunkowe Generowany przez menedżer kolejek. Odpowiedź, Raport Generowane są zestawy nadajnika SOAP IBM MQ MQRO_NEW_MSG_ID i <i>MsgId</i> .	Wygen. Unikalna wartość wygenerowana przez menedżer kolejek
<i>CorrelId</i>	Request, One way, Report MQCI_NONE Odpowiedź, Raport Nadajnik SOAP IBM MQ ustawia MQRO_COPY_MSG_ID_TO_CORREL_ID i program nasłuchujący <i>MsgId</i> z komunikatu żądania.	MQCI_NONE 0
<i>BackoutCount</i>	ALL Nieużywane	0
<i>ReplyToQ</i>	Żądanie Określone przez opcję replyDestination w identyfikatorze URI. Wartością domyślną jest SYSTEM.SOAP.RESPONSE.QUEUE. Odpowiedź, Jeden ze sposobów, Raport Puste pole	

Tabela 184. Ustawienia SOAP MQMD (kontynuacja)

Nazwa pola	Ustawienie	Wartości
<i>ReplyToQMGr</i>	ALL Pole pozostawione puste	Wygenerowane przez menedżer kolejek. Patrz sekcja Kolejka odpowiedzi i menedżer kolejek .
<i>UserIdentifier</i>	Request, One way, Report Puste pole Działanie Zależy od opcji -i <i>passContext</i> dostarczonej do programu nasłuchującego, a także od uprawnień, w ramach których działa nasłuchiwanie.	Request, One way, Report Wygenerowane przez menedżer kolejek; patrz “ UserIdentifier (MQCHAR12) ” na stronie 465 Działanie <i>Zmienna</i>
<i>AccountingToken</i>	ALL MQACT_NONE	MQACT_NONE Pusty łańcuch lub odstęp Ustawione przez menedżer kolejek; patrz “ AccountingToken (MQBYTE32) ” na stronie 420
<i>ApplIdentityData</i>	ALL Brak	Pusty łańcuch lub odstęp ²
<i>PutApplType</i>	ALL MQAT_NO_CONTEXT	MQAT_NO_CONTEXT 0 Wartość wygenerowana przez menedżer kolejek; patrz “ Typ PutAppl(MQLONG) ” na stronie 450.
<i>PutApplName</i>	ALL Brak	Wartość wygenerowana przez menedżer kolejek; patrz “ Nazwa PutAppl(MQCHAR28) ” na stronie 449.
<i>PutDate</i>	ALL Brak	Wartość wygenerowana przez menedżer kolejek; patrz “ PutDate (MQCHAR8) ” na stronie 452.
<i>PutTime</i>	ALL Brak	Wartość wygenerowana przez menedżer kolejek; patrz “ PutTime (MQCHAR8) ” na stronie 453.
<i>ApplOriginData</i>	ALL Brak	Pusty łańcuch lub odstęp ²
<i>GroupId</i>	Request, One way, Report MQGI_NONE Działanie Pole jest kopiowane z komunikatu żądania	Wartości null

Tabela 184. Ustawienia SOAP MQMD (kontynuacja)

Nazwa pola	Ustawienie	Wartości
<i>MsgSeqNumber</i>	Request, One way, Report Nieużywane Działanie Pole jest kopiowane z komunikatu żądania	Wygenerowane przez menedżer kolejek. Patrz sekcja Kolejność fizyczna w kolejce .
<i>Offset</i>	Request, One way, Report Nieużywane Działanie Pole jest kopiowane z komunikatu żądania	0
<i>MsgFlags</i>	Request, One way, Report MQMF_NONE Działanie Pole jest kopiowane z komunikatu żądania	MQMF_NONE 0 Patrz sekcja “MsgFlags (MQLONG)” na stronie 439
<i>OriginalLength</i>	Żądanie, jednokierunkowy, odpowiedź MQOL_UNDEFINED Raport Długość oryginalnego komunikatu żądania	MQOL_UNDEFINED -1
Uwagi: - Symbol ~ reprezentuje pojedynczy pusty znak. - Wartość łańcuch o wartości NULL lub odstępy oznacza łańcuch pusty w języku C oraz puste znaki w innych językach programowania.		

Niestandardowe opcje raportu

Użytkownik może napisać własny nadawca SOAP i użyć go wraz z dostarczonym nasłuchiującym. Zwykle można napisać nadawcę, aby zmienić wybór opcji raportu. Programy nasłuchujące SOAP programu IBM MQ obsługują większość kombinacji opcji raportów, zgodnie z opisem na poniższych listach.

- Opcje raportów obsługiwane przez programy nasłuchujące SOAP produktu IBM MQ :
 - MQRO_EXCEPTION
 - MQRO_EXCEPTION_WITH_DATA
 - MQRO_EXCEPTION_WITH_FULL_DATA
 - MQRO_DEAD_LETTER_Q
 - MQRO_DISCARD_MSG
 - MQRO_NONE
 - MQRO_NEW_MSG_ID
 - MQRO_PASS_MSG_ID
 - MQRO_COPY_MSG_ID_TO_CORREL_ID
 - MQRO_PASS_CORREL_ID
- Opcje raportów obsługiwane przez menedżer kolejek:
 - 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

• Następujące opcje raportu nie są obsługiwane przez obiekty nasłuchiwanie SOAP produktu IBM MQ .

- MQRO_PAN
- MQRO_NAN

Sposób działania obiektów nasłuchiwanie SOAP produktu IBM MQ w odpowiedzi na kombinacje produktów MQRO_EXCEPTION_* i MQRO_DISCARD jest opisany w sekcji Tabela 185 na stronie 1508.

Notacja MQRO_EXCEPTION_* wskazuje użycie opcji MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA lub MQRO_EXCEPTION_WITH_FULL_DATA.

<i>Tabela 185. Zachowanie programu nasłuchującego wynikające z ustawień MQRO_EXCEPTION_* i MQRO_DISCARD</i>		
	MQRO_DISCARD włączone	MQRO_DISCARD Nie włączono
MQRO_EXCEPTION_* włączone	Zachowanie domyślne. Jeśli to konieczne, komunikaty raportu są generowane automatycznie, a oryginalne żądanie zostało odrzucone. Jeśli komunikat raportu nie może zostać zwrócony do kolejki odpowiedzi, komunikat raportu jest wysyłany do kolejki niewystających komunikatów.	Jeśli to konieczne, komunikaty raportu są generowane automatycznie, a oryginalny komunikat jest wysyłany do kolejki niedostarczanej poczty. Jeśli komunikat raportu nie może zostać zwrócony do kolejki odpowiedzi, jest on również wysyłany do kolejki niedostarczonych komunikatów. W takim przypadku istnieją dwa wpisy w kolejce niedostarczanych komunikatów dla żądania, które nie powiodło się.
MQRO_EXCEPTION_* Nie włączono	Komunikaty raportu nie są generowane automatycznie, gdy format przychodzący nie zostanie rozpoznany lub przekroczona zostanie liczba prób wycofania. Komunikat nie jest wysyłany do kolejki niedostarczanych komunikatów. Nie jest zwracane żadne powiadomienie, które klient może sprawdzić, a oryginalny komunikat żądania został utracony.	Komunikaty raportu nie są generowane automatycznie, gdy format przychodzący nie zostanie rozpoznany lub przekroczona zostanie liczba prób wycofania. Pierwotny komunikat żądania jest jednak zapisywany w kolejce niedostarczanych komunikatów, gdy w przeciwnym razie zostanie wygenerowany raport.

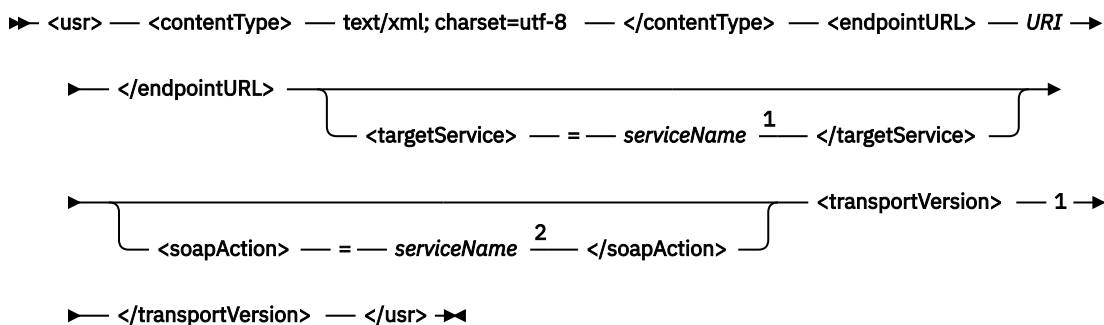
Ustawienia SOAP produktu MQRFH2

Programy nasłuchujące i programy nasłuchujące SOAP produktu IBM MQ tworzą lub oczekują, że otrzymają MQRFH2 z następującymi ustawieniami.

Przeznaczenie

Nadawcy SOAP IBM MQ dodają właściwości do folderu `<usr>` utworzonego przez program IBM MQ JMS. Właściwości zawierają informacje wymagane przez kontener SOAP w środowisku docelowym. [“Składnia właściwości”](#) na stronie 1509 opisuje składnię właściwości, gdy są one dodawane do MQRFH2. Opis nagłówka MQRFH2 znajduje się w sekcji [MQRFH2 -Reguły i formatowanie nagłówka 2](#).

Składnia właściwości



Uwagi:

¹ Usługa targetService jest wymagana dla środowiska .NET w wersji 1 lub 2, a nie jest używana w środowisku Axis 1.4.

² Element soapAction jest opcjonalny w przypadku środowiska .NET Framework 1 lub 2, który nie jest używany w środowisku Axis 1.4.

Parametry

contentType

Element contentType zawsze zawiera łańcuch `text/xml; charset=utf-8`.

endpointURL

Patrz sekcja [“Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service”](#) na stronie 1526.

targetService

⁷W przypadku osi *serviceName* jest to pełna nazwa usługi produktu Java , na przykład: `targetService=javaDemos.service.StockQuoteAxis`. Jeśli parametr `targetService` nie zostanie określony, usługa zostanie załadowana przy użyciu domyślnego mechanizmu Axis.

⁸W systemie .NET *serviceName* jest nazwą usługi .NET , która znajduje się w katalogu wdrażania, na przykład: `targetService=myService.asmx`. W środowisku .NET parametr `targetService` umożliwia pojedynczy program nasłuchujący SOAP produktu IBM MQ , który może przetwarzać żądania wielu usług. Te usługi muszą zostać wdrożone z tego samego katalogu.

soapAction

transportVersion

Parametr `transportVersion` jest zawsze ustawiony na wartość 1.

Przykład

W przykładzie przedstawiono MQRFH2 i następujący komunikat SOAP. Długości folderów są wyświetlane w postaci dziesiętnej.

Uwaga: & w identyfikatorze URI jest kodowane jako `&`

```
52464820 00000002 000002B0 00000001 RFH/ 0002 1208 0001
000004B8 20202020 20202020 00000000 1208 ? ? ? ? ? ? ? 0000
```

⁷ Tylko usługa Java

⁸ Tylko usługa .NET

```

00000488          1208
32 <mcd>
<Msd>jms_bytes</Msd>
</mcd>?
208 <jms>
<Dst>queue://queue://SOAPJ.demos</Dst>
<Rto>queue://WMQSOAP.DEMO.QM/SYSTEM.SOAP.RESPONSE.QUEUE</Rto>
<Tms>1157388516465</Tms>
<Cid>ID:0000000000000000000000000000000000000000000000000000000000000000</Cid>
<Dlv>1</Dlv>
</jms>
400 <usr>
<contentType>text/xml; charset=utf-8</contentType>
<transportVersion>1</transportVersion>
<endpointURL>
jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
&amp;connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)
clientConnection(localhost%25289414%2529)
clientChannel(TESTCHANNEL)
&amp;replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
</endpointURL>
</usr>
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="https://www.w3.org/2001/XMLSchema"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:getQuote
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="soap.server.StockQuoteAxis_Wmq">
<in0 xsi:type="xsd:string">XXX</in0>
</ns1:getQuote>
</soapenv:Body>
</soapenv:Envelope>

```

runivt: transport produktu IBM MQ do sprawdzania poprawności instalacji SOAP

Pakiet testów weryfikacji instalacji (IVT) jest dostarczany z transportem IBM MQ dla protokołu SOAP. Produkt **runivt** uruchamia wiele aplikacji demonstracyjnych i zapewnia, że środowisko jest poprawnie skonfigurowane po instalacji.

Przeznaczenie

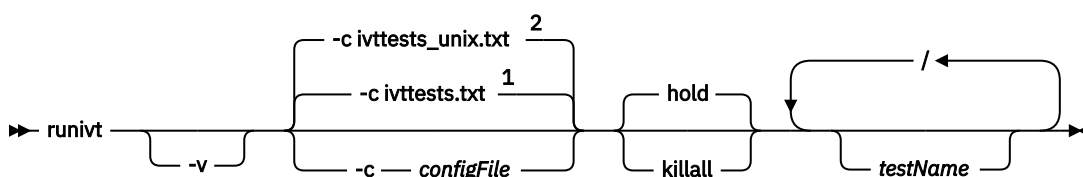
Komenda **runivt** używa przykładowych programów dostarczanych razem z transportem IBM MQ dla protokołu SOAP do wysyłania żądań usług Web Service od klientów do usług. Program ten uruchamia testy dla środowiska Axis 1.4, .NET Framework 1 i .NET Framework 2. Testy są konfigurowane w pliku skryptu testowego. Domyślny plik skryptu testowego dla produktu Windows uruchamia kombinację testów między klientami i usługami Java i .NET .

Opis

Produkt **runivt** musi być uruchamiany z poziomu własnego katalogu.

Komenda uruchamia obiekty nasłuchiwanie w innym oknie komend. Z tego powodu należy uruchomić komendę z sesji X Window System w systemach UNIX i Linux .

runivt Składnia



Uwagi:

¹ Wartość domyślna w systemie Windows

² Wartość domyślna w systemach UNIX i Linux

Parametry runivt

-v

Tryb opisowy Zapisz pełniejsze komunikaty o błędach w konsoli.

-c configFile

Plik konfiguracyjny definiujący testy, które mają zostać uruchomione. Domyślny plik konfiguracyjny dostarczany razem z systemami Windows, UNIX lub Linux jest używany domyślnie.

hold

Pozostaw nasłuchiwanie uruchomione po zakończeniu testów

killall

Zakończ nasłuchiwanie po zakończeniu testów

testName

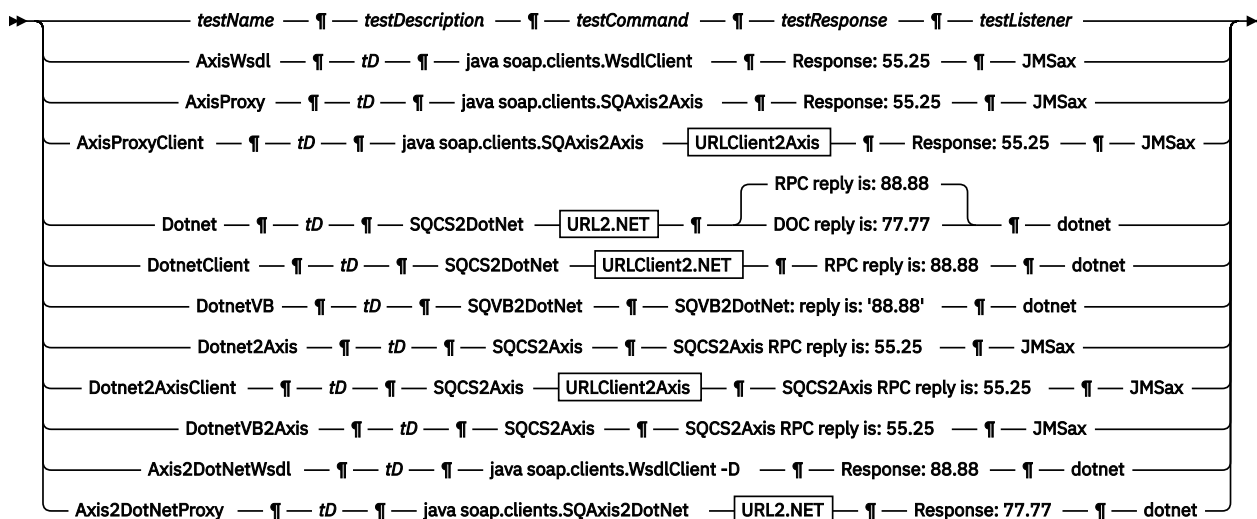
Lista oddzielonych spacjami testów do uruchomienia. Nazwy testów są wybierane z pliku konfiguracyjnego. Jeśli nie zostaną podane żadne nazwy, zostaną uruchomione wszystkie testy w pliku konfiguracyjnym.

Plik konfiguracyjny

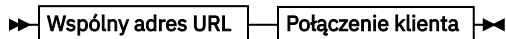
Każdy parametr pliku konfiguracyjnego jest oddzielnym wierszem pliku. Pozostaw pustą linię między poszczególnymi grupami parametrów.

Zostaną wyświetlone parametry podane w pliku parametrów `ivttests.txt`.

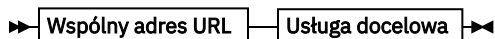
Składnia configFile



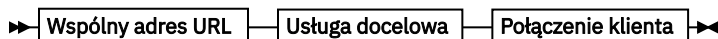
URLClient2Axis



URL2.NET



URLClient2.NET



Wspólny adres URL

```

➤ jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM — & — initialContextFactory — = ➔

    ➤ com.ibm.mq.jms.Nojndi — & — connectionFactory — = ➔

    ➤ connectQueueManager — ( — WMQSOAP.DEMO.QM — ) ➤

```

Połączenie klienta

```

➤ clientConnection — ( — localhost%25289414WMQSOAP.DEMO.QM%2529 — ) — clientChannel ➔

    ➤ ( — TESTCHANNEL — ) ➤

```

Usługa docelowa

```

➤ & — targetService — = — StockQuoteDotNet.asmx ➤

```

Parametry *configFile*

testName

Nazwa testu. Użyj komendy *testName* w komendzie **runivt**.

testDescription

Documentation na temat testu

testCommand

Komenda wykonana przez komendę **runivt** w celu wykonania żądania klienta.

testResponse

Dokładny łańcuch odpowiedzi zwrócony przez żądanie klienta do konsoli. Aby test powiódł się *testResponse*, musi być zgodny z rzeczywistą odpowiedzią.

testListener

Nazwa programu nasłuchującego SOAP produktu IBM MQ, który jest uruchamiany przez produkt **runivt** w celu przetworzenia żądania SOAP. Produkty **dotnet** i **JMSax** są synonimami dla dostarczonych programów nasłuchujących, **amqwSOAPNETlistener** i **SimpleJavaListener**.

Przykłady

```
runivt
```

Rysunek 18. uruchom wszystkie testy domyślne

```
runivt dotnet
```

Rysunek 19. uruchomić konkretny test z testów domyślnych

```
runivt -c mytests.txt
```

Rysunek 20. uruchomienie zestawu testów niestandardowych

Informacje pokrewne

[Weryfikowanie transportu produktu IBM MQ dla protokołu SOAP](#)

Bezpieczne usługi Web Service przy użyciu transportu IBM MQ dla protokołu SOAP

Można zabezpieczyć usługi Web Service, które korzystają z transportu IBM MQ dla protokołu SOAP na jeden z dwóch sposobów. Utwórz kanał SSL między klientem i serwerem lub użyj zabezpieczeń usług Web Service.

Protokół SSL i transport protokołu IBM MQ dla protokołu SOAP

Transport produktu IBM MQ dla protokołu SOAP udostępnia kilka opcji SSL, które można określić dla kanału klienta skonfigurowanego do uruchamiania w trybie SSL. Opcje różnią się między środowiskami .NET i Java. Programy nasłuchujące SOAP i procesy nasłuchujące produktu IBM MQ przetwarzają tylko opcje SSL, które mają zastosowanie do ich środowiska. Ignorują one opcje, które nie mają zastosowania.

Obecność lub brak opcji `sslCipherSpec` dla klientów .NET oraz opcji `sslCipherSuite` dla klientów Java określa, czy używany jest protokół SSL, czy też nie. Jeśli ta opcja nie zostanie podana w identyfikatorze URI, domyślnie nie jest używany protokół SSL, a wszystkie pozostałe opcje protokołu SSL są ignorowane. Wszystkie opcje SSL są opcjonalne, z wyjątkiem przypadków, w których wskazano.

W przypadku klientów IBM MQ ustaw atrybuty SSL w identyfikatorze URI lub tabeli definicji kanału. Na serwerze ustaw atrybuty przy użyciu narzędzi produktu IBM MQ.

Domyślnie opcja `SSLCAUTHw` standardzie IBM MQ SSL jest ustawiana podczas włączania protokołu SSL na kanał. Klienci muszą się uwierzytelnić przed rozpoczęciem komunikacji SSL. Jeśli parametr `SSLCAUTH` nie jest ustawiony, komunikacja SSL jest ustanawiana bez uwierzytelniania klienta.

Aby się uwierzytelnić, klienci muszą posiadać certyfikat przypisany do swojego repozytorium kluczy, który jest akceptowalny dla menedżera kolejek. W celu uzyskania dodatkowych zabezpieczeń można skonfigurować kanały produktu IBM MQ w taki sposób, aby akceptują tylko certyfikaty z listy zastrzeżonych. Lista jest ograniczona, sprawdzając nazwę wyróżniającą certyfikatu dla atrybutu nazwy węzła kanału.

Jeśli używany jest produkt Java, pierwsze połączenie SSL z klienta SOAP IBM MQ spowoduje, że zostaną ustalone następujące parametry SSL. Te same wartości są używane w kolejnych połączeniach przy użyciu tego samego procesu klienta:

- `sslKeyStore`
- `sslKeyStorePassword`
- `sslTrustStore`
- `sslTrustStorePassword`
- `SSLFIPSREQUIRED`
- `sslLDAPCRLservers`

Wpływ zmiennych tych parametrów na kolejne połączenia z tego klienta jest niezdefiniowany.

Jeśli używany jest produkt .NET, pierwsze połączenie SSL z klienta SOAP IBM MQ spowoduje, że zostaną ustalone następujące parametry SSL. Te same wartości są używane w kolejnych połączeniach przy użyciu tego samego procesu klienta:

- `sslKeyRepozytorium`
- `SprzetsslCrypto-sprzet`
- `SSLFIPSREQUIRED`
- `sslLDAPCRLservers`

Wpływ zmiennych tych parametrów na kolejne połączenia z tego klienta jest niezdefiniowany. Te parametry są resetowane, jeśli wszystkie połączenia SSL staną się nieaktywne i zostanie nawiązane nowe połączenie SSL.

Jako właściwości systemowe można również określić następujące właściwości:

- `sslKeyStore`
- `sslKeyStorePassword`
- `sslTrustStore`
- `sslTrustStorePassword`

Jeśli są one określone zarówno jako właściwości systemowe, jak i w identyfikatorze URI, a wartości różnią się, program narzędziowy do wdrażania wyświetli ostrzeżenie. Wartości identyfikatora URI mają pierwszeństwo.

Odsyłacze pokrewne

[Parametry fabryki połączeń SSL w identyfikatorze URI usług Web Service produktu IBM MQ](#)

[Dodaj opcje SSL do listy opcji fabryki połączeń w identyfikatorze URI usług Web Service produktu IBM MQ](#).

Informacje pokrewne

[Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS](#)

[FIPS \(Federal Information Processing Standards\) dla produktów UNIX, Linux i Windows](#)

Parametry fabryki połączeń SSL w identyfikatorze URI usług Web Service produktu IBM MQ

Dodaj opcje SSL do listy opcji fabryki połączeń w identyfikatorze URI usług Web Service produktu IBM MQ.

Przeznaczenie

Istnieje możliwość użycia bezpiecznego połączenia między klientem usług Web Service produktu IBM MQ a menedżerem kolejek udostępniającego usługę Web Service. Opcje protokołu SSL sterują sposobem skonfigurowania protokołu SSL w połączeniu kanału serwera IBM MQ MQI client.

Diagram składni

SSL (Java)

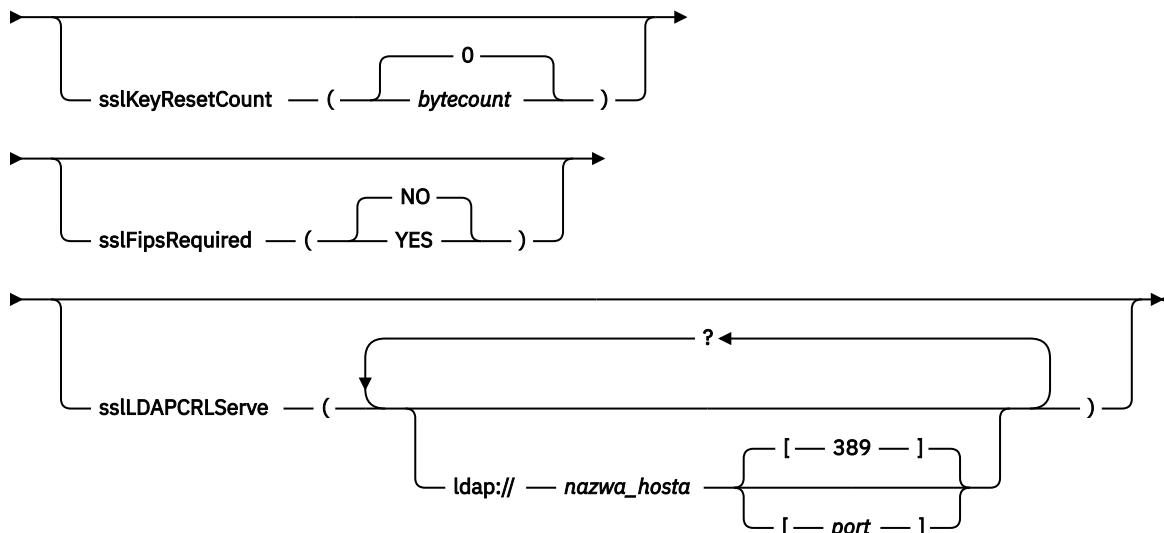
```
➤➤ sslCipherSuite — ( — CipherSuite — ) — sslKeyStore — ( — KeyStoreNazwa — ) ➔➤  
    ➤➤ sslKeyPassword — ( — KeyStoreHasło — ) — sslTrustStore — ( — TrustStore — ) ➔➤  
    ➤➤ sslTrustStorePassword — ( — TrustStoreHasło — ) — SSL (wspólny) ➤➤
```

SSL (.NET)

```
➤➤ sslCipherSpec — ( — CipherSpec — ) — sslKeyRepository — ( — KeyRepository — ) ➔➤  
    ➤➤ CryptoHardware — SSL (wspólny) ➤➤
```

SSL (wspólny)

➤ sslCipherPeerName — (— *PeerName* —) ➔



CryptoHardware

➤ sslCryptoHardware — = — *PKCS #11 Ścieżka i nazwa pliku* — ; — *Etykieta tokenu PKCS #11* — ; ➔

➤ *Hasło tokenu PKCS #11* — ; — *ustawienie szyfru symetrycznego* — ; ➔

Wymagane parametry SSL (wspólne)

sslPeerName (*peerName*)

Parametr *peerName* określa nazwę sslPeerName używaną na kanale.

Wymagane parametry SSL (Java)

sslCipherSuite (*CipherSuite*)

CipherSuite określa parametr sslCipherSuite używany na kanale. Opcja CipherSuite określona przez klienta musi być zgodna z zestawem CipherSuite określonym w kanale połączenia serwera.

sslKeyStore (*KeyStoreNazwa*)

KeyStoreNazwa określa wartość sslKeyStoreName używaną na kanale. Magazyn kluczy przechowuje klucz prywatny klienta używanego do uwierzytelniania klienta na serwerze. Magazyn kluczy jest opcjonalny, jeśli połączenie SSL zostało skonfigurowane w taki sposób, aby akceptować anonimowe połączenia klienta.

sslKeyStorePassword (*KeyStoreHasło*)

KeyStoreHasło określa wartość sslKeyStorePassword używaną na kanale.

sslTrustStore (*TrustStoreNazwa*)

TrustStoreNazwa określa sslTrustStoreName używany na kanale. Magazyn zaufanych certyfikatów przechowuje publiczny certyfikat serwera lub jego łańcuch kluczy w celu uwierzytelnienia serwera na kliencie. Magazyn zaufanych certyfikatów jest opcjonalny, jeśli do uwierzytelniania serwera używany jest certyfikat główny ośrodka certyfikacji. W produkcie Javacertyfikaty główne są przechowywane w bazie certyfikatów środowiska JRE (cacerts).

sslTrustStorePassword (*TrustStoreHasło*)

TrustStoreHasło określa parametr sslTrustStorePassword używany na kanale.

Wymagane parametry SSL (.NET)

sslCipherSpec (*CipherSpec*)

CipherSpec określa wartość sslCipherSpec używaną na kanale. Jeśli ta opcja jest określona, w kanale klienta jest używany protokół SSL.

sslKeyRepository (*KeyRepository*)

KeyRepository określa wartość sslCipherSpec używaną w kanale, w którym przechowywane są klucze i certyfikaty SSL. Opcja *KeyRepository* jest określona w formacie macierzystym, to znaczy pełna ścieżka z nazwą pliku, ale z pominięciem rozszerzenia pliku. Ustawienie właściwości sslKeyRepository jest takie samo, jak ustawienie pola KeyRepository w strukturze **MQSCO** w wywołaniu MQCONN.

Opcjonalne parametry SSL (.NET)

sslCryptoHardware (*CryptoHardware*)

CryptoHardware określa sslCryptoHardware (Sprzęt) używany na kanale. Możliwe wartości dla tego pola oraz efekt jego ustawienia są takie same, jak w przypadku pola CryptoHardware struktury **MQSCO** na serwerze MQCONN.

Opcjonalne parametry SSL (Common)

sslKeyResetCount (*bytecount*)

Parametr *bytecount* określa liczbę bajtów przekazywanych przez kanał SSL przed ponownym negocjacją klucza tajnego SSL. Aby wyłączyć renegotiację kluczy SSL, pomiń pole lub ustaw wartość zero. Wartość zero jest jedyną wartością obsługiwaną w niektórych środowiskach. Patrz sekcja [Renegotiowanie klucza tajnego w produkcie IBM MQ classes for Java](#). Efekt ustawienia sslKeyResetCount jest taki sam, jak ustawienie pola KeyResetCount w strukturze **MQSCO** w wywołaniu MQCONN.

sslFipsRequired (*fipsCertified*)

fipsCertified określa, czy parametr *CipherSpec* lub *CipherSuite* musi używać kryptografii z certyfikatem FIPS w produkcie IBM MQ na kanale. Ustawienie parametru *fipsCertified* jest takie samo, jak ustawienie pola FipsRequired struktury **MQSCO** w wywołaniu MQCONN.

sslLDAPCRLServers (*LDAPServerList*)

LDAPServerList określa listę serwerów LDAP, które mają być używane na potrzeby sprawdzania listy odwołań certyfikatów.

W przypadku połączeń klienckich z obsługą SSL *LDAPServerList* to lista serwerów LDAP, które mają być używane do sprawdzania listy CRL (Certificate Revocation List). Certyfikat udostępniony przez menedżer kolejek jest sprawdzany w odniesieniu do jednego z wymienionych serwerów CRL LDAP. Jeśli zostanie znaleziony, połączenie nie powiedzie się. Każdy serwer LDAP jest podejmowany z kolei do czasu nawiązania połączenia z jednym z nich. Jeśli nawiązanie połączenia z żadnym z serwerów nie jest możliwe, certyfikat zostanie odrzucony. Po pomyślnym nawiązaniu połączenia z jednym z nich certyfikat jest akceptowany lub odrzucany w zależności od list CRL znajdujących się na tym serwerze LDAP.

Jeśli pole *LDAPServerList* jest puste, certyfikat należący do menedżera kolejek nie jest sprawdzany na liście odwołań certyfikatów. Jeśli podana lista identyfikatorów URI LDAP nie jest poprawna, zostanie wyświetlony komunikat o błędzie. Ustawienie tego pola jest takie samo, jak w przypadku rekordów MQAIR i uzyskiwanie dostępu do nich ze struktury **MQSCO** na serwerze MQCONN.

Odsyłacze pokrewne

Protokół SSL i transport protokołu IBM MQ dla protokołu SOAP

Transport produktu IBM MQ dla protokołu SOAP udostępnia kilka opcji SSL, które można określić dla kanału klienta skonfigurowanego do uruchamiania w trybie SSL. Opcje różnią się między środowiskami .NET i Java. Programy nasłuchujące SOAP i procesy nasłuchujące produktu IBM MQ przetwarzają tylko opcje SSL, które mają zastosowanie do ich środowiska. Ignorują one opcje, które nie mają zastosowania.

Informacje pokrewne

Określanie, że w czasie wykonywania w kliencie MQI są używane tylko specyfikacje CipherSpecs z certyfikatem FIPS

FIPS (Federal Information Processing Standards) dla produktów UNIX, Linux i Windows

SimpleJavaListener: program nasłuchujący SOAP IBM MQ dla osi 1.4

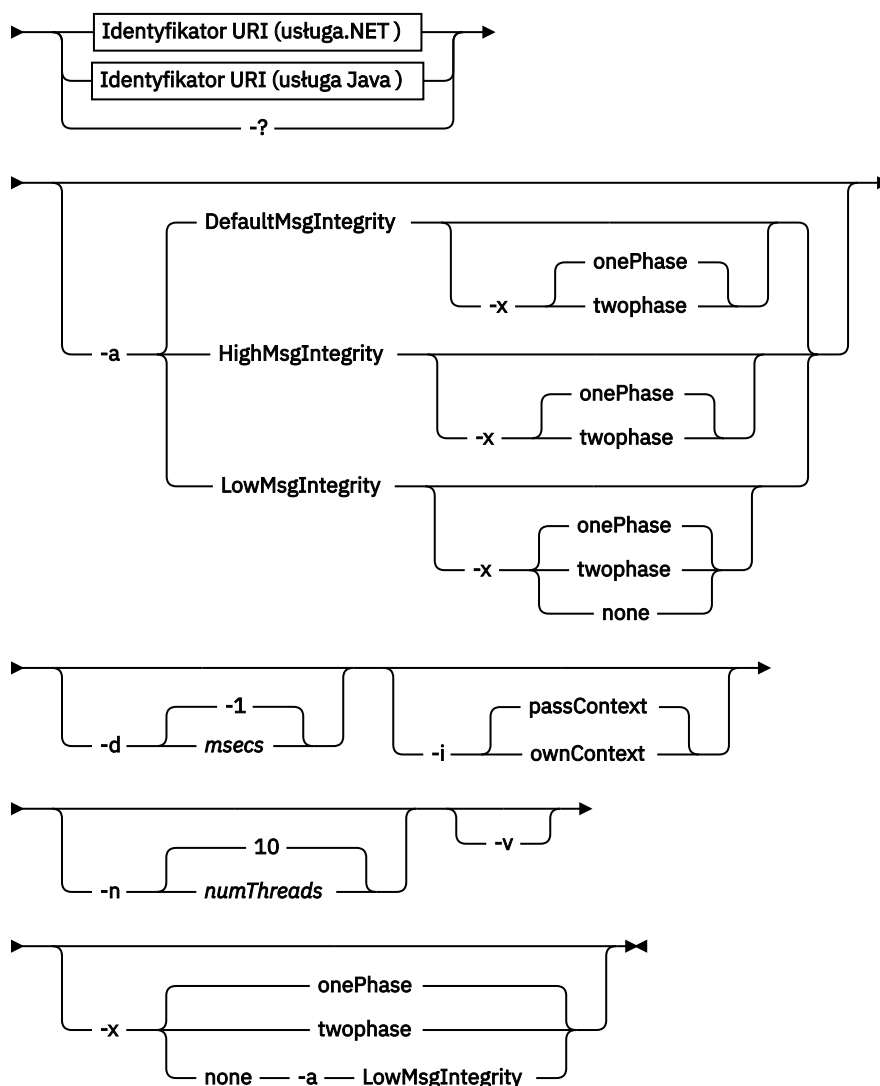
Składnia i parametry dla programu nasłuchującego SOAP produktu IBM MQ dla osi 1.4.

Przeznaczenie

Uruchamia program nasłuchujący SOAP produktu IBM MQ dla osi 1.4.

Java

►► java — com.ibm.mq.soap.transport.jms.SimpleJavaListener —►



Wymagane parametry

URI platforma

Patrz ["Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service"](#) na stronie 1526.

-?

Wydrukuj tekst pomocy opisujący sposób użycia komendy.

Parametry opcjonalne

-a *integrityOption*

Opcja *integrityOption* określa zachowanie obiektów nasłuchiwania SOAP produktu IBM MQ , jeśli nie jest możliwe umieszczenie komunikatu żądania, który nie powiódł się w kolejce niedostarczonych komunikatów. *integrityOption* może przyjmować jedną z następujących wartości:

DefaultMsgIntegrity

W przypadku komunikatów nietrwałych program nasłuchujący wyświetla komunikat ostrzegawczy i kontynuuje wykonywanie, gdy oryginalny komunikat jest odrzucany. W przypadku komunikatów trwałych wyświetlany jest komunikat o błędzie, który powoduje utworzenie kopii zapasowej komunikatu żądania w taki sposób, aby pozostał on w kolejce żądań i kończy działanie.

DefaultMsgIntegralność ma zastosowanie, jeśli pominięto opcję -a lub jeśli opcja *integrityOption* nie została określona.

LowMsgIntegrity

Zarówno dla komunikatów trwałych, jak i nietrwałych program nasłuchujący wyświetla ostrzeżenie i kontynuuje wykonywanie, odrzucając komunikat.

HighMsgIntegrity

W przypadku komunikatów trwałych i nietrwałych program nasłuchujący wyświetla komunikat o błędzie, tworzy kopię zapasową komunikatu żądania w taki sposób, że pozostaje w kolejce żądań i kończy działanie.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Jeśli zostanie podana wartość -x none , należy podać wartość -a LowMsgIntegrity . Jeśli flagi są niezgodne, program narzędziowy wdrażania zakończy działanie z komunikatem o błędzie i nie zostało wykonane żadne kroki wdrażania.

-d *msecs*

msecs określa liczbę milisekund dla programu nasłuchującego SOAP produktu IBM MQ , który ma pozostać przy życiu, jeśli komunikaty żądania zostały odebrane w dowolnym wątku. Jeśli parametr *msecs* ma wartość -1, program nasłuchujący pozostanie aktywny na czas nieokreślony.

-i *Kontekst*

Kontekst określa, czy obiekty nasłuchiwania przekazują kontekst tożsamości. *Kontekst* przyjmuje następujące wartości:

passContext

Ustaw kontekst tożsamości oryginalnego komunikatu żądania w komunikacie odpowiedzi. Program nasłuchujący SOAP sprawdza, czy ma uprawnienia do zapisywania kontekstu z kolejki żądań i przekazywania go do kolejki odpowiedzi. Podczas uruchamiania kolejki żądań w celu zapisania kontekstu oraz kolejki odpowiedzi do przekazywania kontekstu są wykonywane sprawdzanie w czasie wykonywania. Jeśli nie ma wymaganych uprawnień lub wywołanie MQOPEN nie powiedzie się, a komunikat odpowiedzi nie zostanie przetworzony. Komunikat odpowiedzi jest umieszczany w kolejce niedostarczonych komunikatów z nagłówkiem niedostarczonych komunikatów, który zawiera kod powrotu z uszkodzonego MQOPEN. Następnie program nasłuchujący kontynuuje przetwarzanie kolejnych komunikatów przychodzących w normalny sposób.

ownContext

Obiekt nasłuchiwania SOAP nie przekazuje kontekstu. Zwrócony kontekst odzwierciedla ID użytkownika, pod którym nasłuchiwanie jest uruchomione, a nie identyfikator użytkownika, który utworzył oryginalny komunikat żądania.

Pola w kontekście pochodzenia są ustawiane przez menedżer kolejek, a nie przez obiekt nasłuchiwania SOAP.

-n *numThreads*

numThreads określa liczbę wątków w wygenerowanym skrypcie startowym dla programu nasłuchującego SOAP produktu IBM MQ . Wartość domyślna wynosi 10. Jeśli przepustowość komunikatów jest wysoka, należy rozważyć zwiększenie tej liczby.

-v

-v ustawia szczegółowe dane wyjściowe z komend zewnętrznych. Komunikaty o błędach są zawsze wyświetlane. Komenda -v służy do wyprowadzania komend, które można dostosować do niestandardowych skryptów wdrażania.

-w **serviceDirectory**

serviceDirectory to katalog zawierający usługę Web Service.

-x **transakcyjność**

transakcyjność określa typ sterowania transakcyjnego dla obiektu nasłuchiwanego. *transakcyjność* można ustawić na jedną z następujących wartości:

onePhase

IBM MQ jest używana obsługa jednofazowa. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania zostanie ponownie dostarczony do aplikacji. Transakcje IBM MQ zapewniają, że komunikaty odpowiedzi są zapisywane dokładnie jeden raz.

twoPhase

Używana jest obsługa dwufazowa. Jeśli usługa jest odpowiednio napisana, komunikat jest dostarczany dokładnie jeden raz, koordynowany z innymi zasobami, w ramach pojedynczego zatwierdzonego wykonania usługi. Ta opcja ma zastosowanie tylko do połączeń z powiązaniami serwera.

none

Brak obsługi transakcyjnej. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania może zostać utracony, nawet jeśli jest on trwały. Możliwe, że usługa została wykonana lub nie została wykonana, a odpowiedź, raport lub komunikaty o niedostarczonych literach mogą, ale nie, być zapisywane.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Szczegółowe informacje można znaleźć w opisie opcji -a .

Java Przykład

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.NoJndi"
-n 20
```

Obiekty nasłuchiwania SOAP produktu IBM MQ

Program nasłuchujący SOAP IBM MQ odczytuje przychodzące żądanie SOAP z kolejki określonej jako miejsce docelowe w identyfikatorze URI. Sprawdza on format komunikatu żądania, a następnie wywołuje usługę Web Service przy użyciu infrastruktury usług Web Service. Program nasłuchujący SOAP IBM MQ zwraca dowolną odpowiedź lub błąd z usługi Web Service przy użyciu kolejki miejsca docelowego odpowiedzi w identyfikatorze URI. Zwraca on raporty IBM MQ do kolejki odpowiedzi.

Ten obiekt nasłuchiwania terminów jest używany w jego standardowym sensie usług WWW. Różni się on od standardowego programu nasłuchującego IBM MQ wywoływanego przez komendę **runmq1sr** .

Opis

Program nasłuchujący SOAP produktu Java jest implementowany jako klasa produktu Java , a usługi uruchamiania za pomocą środowiska Axis 1.4. Program nasłuchujący .NET jest aplikacją konsoli i uruchamia usługi .NET Framework 1 lub .NET Framework 2. W przypadku usług .NET Framework 3 należy użyć niestandardowego kanału IBM MQ dla programu Microsoft Windows Communication Foundation (WCF).

Program narzędziowy do wdrażania tworzy skrypty w celu automatycznego uruchamiania programów nasłuchujących SOAP produktu Java lub .NET . Program nasłuchujący SOAP można uruchomić ręcznie za pomocą komendy **amqSOAPNETListener** lub wywołując klasę **SimpleJavaListener** . Usługę nasłuchiwaną SOAP produktu IBM MQ można skonfigurować w taki sposób, aby była uruchamiana

jako usługa IBM MQ, ustawiając opcję -s w programie narzędziowym wdrażania. Można również uruchomić obiekty nasłuchiwanie przy użyciu wyzwalania lub użyć skryptów programu nasłuchującego uruchamiania i zakończenia wygenerowanych przez program narzędziowy do wdrażania. Można skonfigurować wyzwalanie ręcznie lub użyć opcji wdrażania -tmq i -tmp w celu automatycznego skonfigurowania wyzwalania. Proces nasłuchiwanie można zakończyć, ustawiając kolejkę żądań na wartość GET (DISABLED).

Tabela 186. Skrypty komend wygenerowane przez program narzędziowy do wdrażania

Infrastruktura usług Web Service	Systemy UNIX i Linux	Windows Java	Windows.NET
Uruchom proces nasłuchujący	startWMQJListener.sh	startWMQJListener.cmd	startWMQNListener.cmd
Zatrzymaj proces nasłuchujący	endWMQJListener.sh	endWMQJListener.cmd	endWMQNListener.cmd
Zdefiniuj usługę nasłuchiwanie	defineWMQJListener.sh	defineWMQJListener.cmd	defineWMQNListener.cmd

Program nasłuchujący SOAP IBM MQ przekazuje pola endpointURL i soapAction z komunikatu SOAP do infrastruktury SOAP. Program nasłuchujący wywołuje usługę za pośrednictwem infrastruktury usług Web Service i oczekuje na odpowiedź. Program nasłuchujący nie sprawdza poprawności endpointURL i soapAction. Pola są ustawiane przez nadawcę SOAP IBM MQ na podstawie danych podanych w identyfikatorze URI ustawionym przez klienta SOAP.

Program nasłuchujący tworzy komunikat odpowiedzi i wysyła go do miejsca docelowego odpowiedzi dostarczanego w identyfikatorze URI komunikatu żądania. Dodatkowo obiekt nasłuchiwanie ustawia identyfikator korelacji w komunikacie odpowiedzi zgodnie z opcją raportu w komunikacie żądania. Zwraca on ustawienia utraty ważności, trwałości i priorytetu z komunikatu żądania. Program nasłuchujący wysyła również komunikaty raportów z powrotem do klientów w pewnych okolicznościach.

Jeśli w żądaniu SOAP wystąpią błędy formatowania, program nasłuchujący zwraca komunikat raportu do klienta, korzystając z kolejki docelowej odpowiedzi. Menedżer kolejek zwraca również komunikaty raportów do klienta, korzystając z kolejki miejsca docelowego odpowiedzi, jeśli zażądano raportu. Pełne komunikaty raportu są zapisywane do kolejki odpowiedzi w odpowiedzi na kilka zdarzeń:

- Wyjątek.
- Utrata ważności komunikatu.
- Format komunikatu żądania nie został rozpoznany.
- Sprawdzenie integralności nagłówka **MQRFH2** nie powiodło się.
- Format głównego treści komunikatu nie jest następujący: MQFMT_NONE.
- Próg wycofania/ponowienia jest przekroczony, podczas gdy program nasłuchujący SOAP IBM MQ przetwarza żądanie.

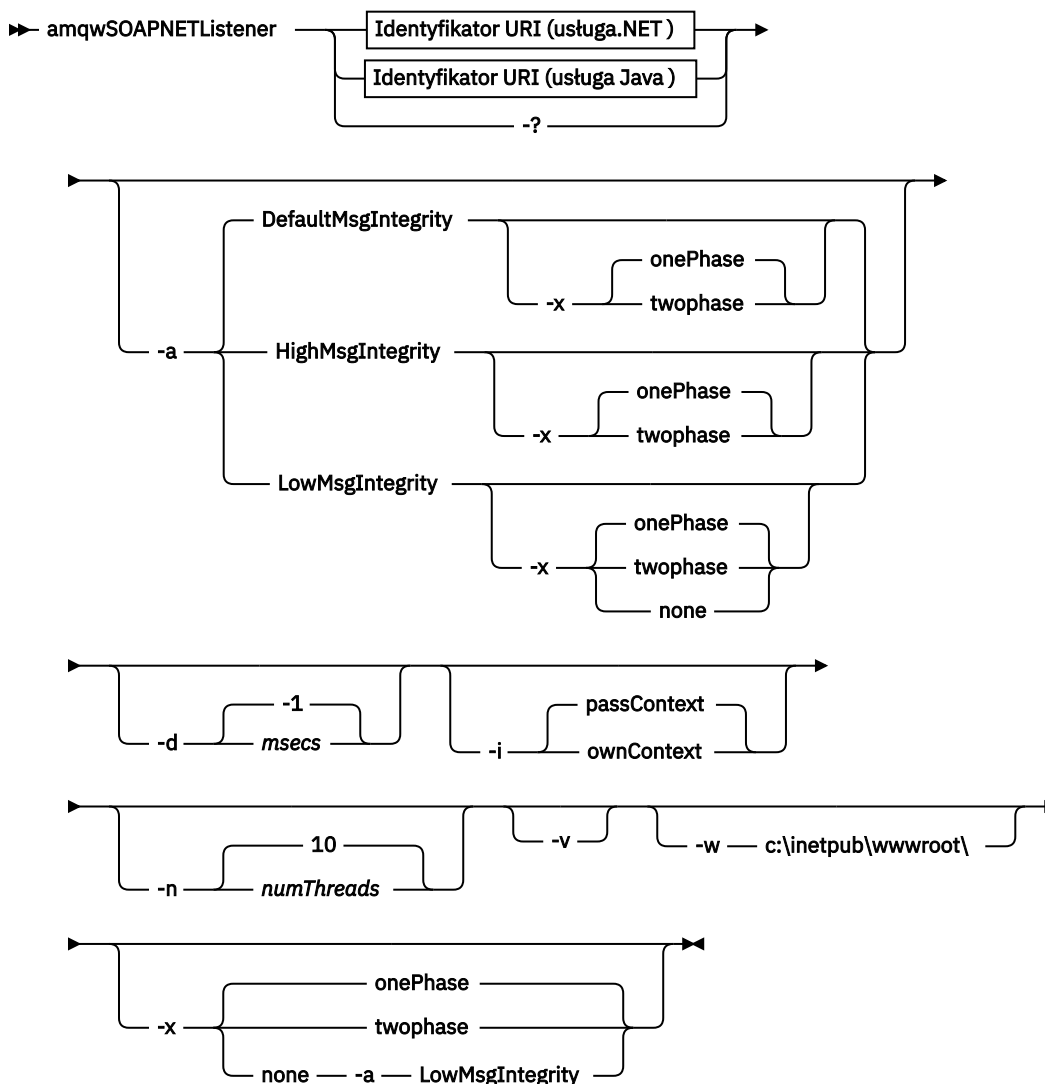
Nadawnik SOAP IBM MQ ustawia opcje raportu MQRO_EXCEPTION_WITH_FULL_DATA i MQRO_EXPIRATION_WITH_FULL_DATA. W wyniku opcji raportu ustawionych przez nadawcę SOAP IBM MQ komunikat raportu zawiera cały komunikat inicjujący żądanie. Nadawca SOAP IBM MQ ustawia również opcję MQRO_DISCARD, która powoduje, że komunikat zostanie usunięty po zwróconej komunikacie raportu. Jeśli opcje raportu nie spełniają wymagań użytkownika, należy napisać własne nadawcy, aby użyć różnych opcji raportu MQRO_EXCEPTION i MQRO_DISCARD. Jeśli żądanie SOAP jest wysyłane przez innego nadawcę, który nie ustawił wartości MQRO_DISCARD, komunikat o niepowodzeniu jest zapisywany w kolejce niedostarczanych komunikatów (DLQ).

Jeśli program nasłuchujący wygeneruje komunikat raportu, ale kończy się niepowodzeniem w procesie wysyłania raportu, komunikat raportu jest wysyłany do kolejki DLQ. Upewnij się, że procedura obsługi DLQ obsługuje te komunikaty poprawnie.

Jeśli wystąpi błąd podczas próby zapisu do kolejki niedostarczonych komunikatów, komunikat jest zapisywany w dzienniku błędów systemu IBM MQ. To, czy program nasłuchujący kontynuuje

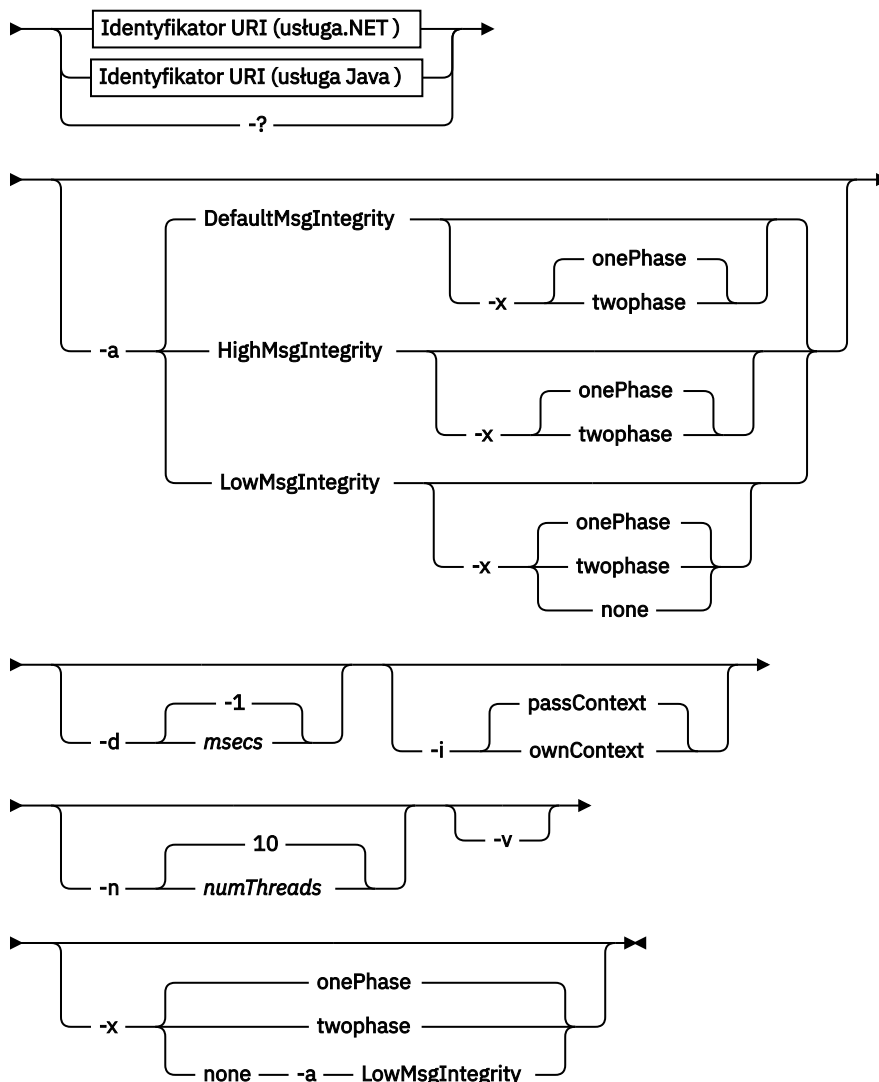
przetwarzanie większej liczby komunikatów, zależy od tego, która trwałość komunikatu i opcje transakcyjne są wybrane. Jeśli program nasłuchujący jest uruchomiony w trybie transakcyjnym jednofazowym i przetwarza komunikat żądania nietrwałego, oryginalny komunikat jest odrzucany. Proces nasłuchiwanie SOAP produktu IBM MQ jest kontynuowany. Jeśli komunikat żądania jest trwały, komunikat żądania jest wycofany do kolejki żądań, a program nasłuchujący kończy działanie. Kolejka żądań jest ustawiona w taki sposób, aby zapobiec przypadkowemu wyzwoleniu restartu.

Diagram składni .NET



Java

➡ java — com.ibm.mq.soap.transport.jms.SimpleJavaListener ➡



Wymagane parametry

URI platforma

Patrz [“Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service”](#) na stronie 1526.

-?

Wydrukuj tekst pomocy opisujący sposób użycia komendy.

Parametry opcjonalne

-a integrityOption

Opcja *integrityOption* określa zachowanie obiektów nasłuchiwania SOAP produktu IBM MQ , jeśli nie jest możliwe umieszczenie komunikatu żądania, który nie powiódł się w kolejce niedostarczonych komunikatów. *integrityOption* może przyjmować jedną z następujących wartości:

DefaultMsgIntegrity

W przypadku komunikatów nietrwałych program nasłuchujący wyświetla komunikat ostrzegawczy i kontynuuje wykonywanie, gdy oryginalny komunikat jest odrzucany. W przypadku komunikatów trwałych wyświetlany jest komunikat o błędzie, który powoduje utworzenie kopii zapasowej komunikatu żądania w taki sposób, aby pozostał on w kolejce żądań i kończy działanie.

DefaultMsgIntegralność ma zastosowanie, jeśli pominięto opcję -a lub jeśli opcja *integrityOption* nie została określona.

LowMsgIntegrity

Zarówno dla komunikatów trwałych, jak i nietrwałych program nasłuchujący wyświetla ostrzeżenie i kontynuuje wykonywanie, odrzucając komunikat.

HighMsgIntegrity

W przypadku komunikatów trwałych i nietrwałych program nasłuchujący wyświetla komunikat o błędzie, tworzy kopię zapasową komunikatu żądania w taki sposób, że pozostaje w kolejce żądań i kończy działanie.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Jeśli zostanie podana wartość -x none , należy podać wartość -a LowMsgIntegrity . Jeśli flagi są niezgodne, program narzędziowy wdrażania zakończy działanie z komunikatem o błędzie i nie zostało wykonane żadne kroki wdrażania.

-d msec

msec określa liczbę milisekund dla programu nasłuchującego SOAP produktu IBM MQ , który ma pozostać przy życiu, jeśli komunikaty żądania zostały odebrane w dowolnym wątku. Jeśli parametr *msec* ma wartość -1, program nasłuchujący pozostanie aktywny na czas nieokreślony.

-i Kontekst

Kontekst określa, czy obiekty nasłuchiwanie przekazują kontekst tożsamości. *Kontekst* przyjmuje następujące wartości:

passContext

Ustaw kontekst tożsamości oryginalnego komunikatu żądania w komunikacie odpowiedzi. Program nasłuchujący SOAP sprawdza, czy ma uprawnienia do zapisywania kontekstu z kolejki żądań i przekazywania go do kolejki odpowiedzi. Podczas uruchamiania kolejki żądań w celu zapisania kontekstu oraz kolejki odpowiedzi do przekazywania kontekstu są wykonywane sprawdzanie w czasie wykonywania. Jeśli nie ma wymaganych uprawnień lub wywołanie MQOPEN nie powiedzie się, a komunikat odpowiedzi nie zostanie przetworzony. Komunikat odpowiedzi jest umieszczany w kolejce niedostarczonych komunikatów z nagłówkiem niedostarczonych komunikatów, który zawiera kod powrotu z uszkodzonego MQOPEN. Następnie program nasłuchujący kontynuuje przetwarzanie kolejnych komunikatów przychodzących w normalny sposób.

ownContext

Obiekt nasłuchiwanie SOAP nie przekazuje kontekstu. Zwrócony kontekst odzwierciedla ID użytkownika, pod którym nasłuchiwanie jest uruchomione, a nie identyfikator użytkownika, który utworzył oryginalny komunikat żądania.

Pola w kontekście pochodzenia są ustawiane przez menedżer kolejek, a nie przez obiekt nasłuchiwanie SOAP.

-n numThreads

numThreads określa liczbę wątków w wygenerowanym skrypcie startowym dla programu nasłuchującego SOAP produktu IBM MQ . Wartość domyślna wynosi 10. Jeśli przepustowość komunikatów jest wysoka, należy rozważyć zwiększenie tej liczby.

-v

-v ustawia szczegółowe dane wyjściowe z komend zewnętrznych. Komunikaty o błędach są zawsze wyświetlane. Komenda -v służy do wyprowadzania komend, które można dostosować do niestandardowych skryptów wdrażania.

-w serviceDirectory

serviceDirectory to katalog zawierający usługę Web Service.

-x transakcyjność

transakcyjność określa typ sterowania transakcyjnego dla obiektu nasłuchiwanie. *transakcyjność* można ustawić na jedną z następujących wartości:

onePhase

IBM MQ jest używana obsługa jednofazowa. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania zostanie ponownie dostarczony do aplikacji. Transakcje IBM MQ zapewniają, że komunikaty odpowiedzi są zapisywane dokładnie jeden raz.

twoPhase

Używana jest obsługa dwufazowa. Jeśli usługa jest odpowiednio napisana, komunikat jest dostarczany dokładnie jeden raz, koordynowany z innymi zasobami, w ramach pojedynczego zatwierdzonego wykonania usługi. Ta opcja ma zastosowanie tylko do połączeń z powiązaniami serwera.

none

Brak obsługi transakcyjnej. Jeśli system nie powiedzie się podczas przetwarzania, komunikat żądania może zostać utracony, nawet jeśli jest on trwały. Możliwe, że usługa została wykonana lub nie została wykonana, a odpowiedź, raport lub komunikaty o niedostarczonych literach mogą, ale nie, być zapisywane.

Program narzędziowy do wdrażania sprawdza kompatybilność opcji -x i -a . Szczegółowe informacje można znaleźć w opisie opcji -a .

.NET Przykład

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.NoJndi"
-w C:/wmqsoap/demos
-n 20
```

Java Przykład

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.NoJndi"
-n 20
```

Transport produktu IBM MQ dla nadawcy SOAP

Klasy nadawcy są udostępniane dla środowiska Axis i środowiska .NET Framework 1 oraz .NET Framework 2. Nadawca konstruuje żądanie SOAP i umieszcza je w kolejce, a następnie blokuje do momentu, gdy odczyta odpowiedź z kolejki odpowiedzi. Zachowanie klas można zmienić, przekazując różne identyfikatory URI z klienta SOAP. W przypadku środowiska .NET Framework 3 należy użyć niestandardowego kanału IBM MQ dla programu Microsoft Windows Communication Foundation (WCF).

Przeznaczenie

Nadawca SOAP IBM MQ umieszcza żądanie SOAP w celu wywołania usługi Web Service w kolejce żądań IBM MQ . Nadawca ustawia pola w nagłówku **MQRFH2** zgodnie z opcjami określonymi w identyfikatorze URI lub zgodnie z wartościami domyślnymi.

Jeśli zachodzi potrzeba zmiany zachowania nadawcy poza tym, co jest możliwe przy użyciu opcji identyfikatora URI, należy napisać własny nadawca. Nadawca może pracować z transportem IBM MQ dla programów nasłuchujących SOAP lub z innymi środowiskami SOAP. Nadawca musi skonstruować komunikaty SOAP w formacie zdefiniowanym przez program IBM MQ. Format jest obsługiwany przez program nasłuchujący SOAP produktu IBM MQ , a także obiekty nasłuchiwanie SOAP udostępniane przez produkt WebSphere Application Server i produkt CICS. Nadawca musi stosować się do reguł dla requestera IBM MQ . Program nasłuchujący SOAP IBM MQ zwraca komunikaty odpowiedzi i raporty. Szczegółowe informacje na temat ustawiania opcji raportu w programie **MQMD** zawiera sekcja ["Ustawienia SOAP produktu MQMD"](#) na stronie 1502 . Opcje raportu sterują komunikatami raportu zwróconego przez program nasłuchujący SOAP IBM MQ .

Opis

The IBM MQ SOAP Java sender is registered with the Axis host environment for the `jms: URI` prefix. Nadawca jest implementowany w klasie `com.ibm.mq.soap.transport.jms.WMQSender`, która pochodzi z produktu `org.apache.axis.handlers.BasicHandler`. Jeśli środowisko hosta Axis wykryje przedrostek identyfikatora URI produktu `jms:` , wywołuje on klasę `com.ibm.mq.soap.transport.jms.WMQSender`. Bloki klas po umieszczeniu komunikatu do momentu odczytania odpowiedzi z kolejki odpowiedzi. Jeśli odpowiedź nie zostanie odebrana w okresie limitu czasu, nadawca zgłosi wyjątek. Jeśli odpowiedź zostanie odebrana w okresie limitu czasu, komunikat odpowiedzi jest zwracany do klienta przy użyciu środowiska Axis. Aplikacja kliencka musi być w stanie obsłużyć te komunikaty odpowiedzi.

W przypadku usług Microsoft.NET Framework 1 i .NET Framework 2 nadawca SOAP IBM MQ jest implementowany w klasie `IBM.WMQSOAP.MQWebRequest`, która jest uzyskiwana z produktów `System.Net.WebRequest` i `System.Net.WebRequest.Create`. Jeśli środowisko .NET Framework 1 lub .NET Framework 2 wykryje przedrostek identyfikatora URI `jms:` , powoduje to wywołanie klasy `IBM.WMQSOAP.MQWebRequest`. Nadawca tworzy obiekt `MQWebResponse` w celu odczytania komunikatu odpowiedzi z kolejki odpowiedzi i zwrócenia go do klienta.

`com.ibm.mq.soap.transport.jms.WMQSender` jest klasą końcową, a `IBM.WMQSOAP.MQWebRequest` jest zabezpieczony. Nie można modyfikować ich zachowania przez tworzenie podklas.

Parametry

Ustaw identyfikator URI, aby sterować zachowaniem nadawcy SOAP produktu IBM MQ w kliencie SOAP usługi Web Service. Program narzędziowy do wdrażania tworzy kody pośredniczące klienta usługi Web Service zawierające opcje identyfikatora URI dostarczone do programu narzędziowego do wdrażania.

Użyj tabeli definicji kanału z transportem SOAP IBM MQ dla nadajnika SOAP

Definicja kanału połączenia klienckiego jest alternatywą do ustawiania właściwości połączenia w atrybucie `ConnectionFactory` identyfikatora URI usługi Web Service. Właściwości połączenia to `clientChannel`, `clientConnection` i `SSL`.

Opis

Utwórz tabelę opisu kanału klienta, definiując połączenia klienckie. Nawet jeśli klient usług Web Service łączy się z różnymi menedżerami kolejek, należy utworzyć wszystkie połączenia w tabeli połączeń w pojedynczym menedżerze kolejek. Domyślna nazwa i położenie tabeli połączeń to *queue manager directory/@ipcc/AMQCLCHL.TAB*.

Przekaz położenia tabeli połączeń do klienta Java, ustawiając właściwość systemową `com.ibm.mq.soap.transport.jms.mqchlurl`.

Przekaz położenie tabeli połączeń do klienta .NET, ustawiając zmienne środowiskowe `MQCHLLIB` i `MQCHLTAB`.

W atrybucie `ConnectionFactory` identyfikatora URI usługi Web Service można podać zarówno tabelę połączeń kanału, jak i parametry połączenia kanału. Wartości ustawione w elemencie `ConnectionFactory` mają pierwszeństwo przed wartościami w tabeli definicji kanału.

Korzystanie z tabeli definicji kanału w produkcji Java

```
java -Dcom.ibm.msg.client.config.location=file:/C:/mydir/myjms.config MyAppClass
```

Rysunek 21. Uruchamianie klienta Java przy użyciu pliku konfiguracyjnego

```
com.ibm.mq.soap.transport.jms.mqchlurl=file:/C:/ibm/wmq/qmgrs/QM1/@ipcc/AMQCLCHL.TAB
```

Rysunek 22. myjms.config

Transakcje

Użyj opcji -x podczas uruchamiania programu nasłuchującego, aby uruchamiać usługi Web Service. Aby wybrać integralność komunikatów, należy ustawić opcję trwałość w identyfikatorze URI usługi.

Usługi WWW

Użyj opcji -x podczas uruchamiania programu nasłuchującego, aby uruchamiać usługi Web Service. W środowisku .NET Framework 1 i 2 program nasłuchujący SOAP korzysta z programu Microsoft Transaction Coordinator (MTS). W przypadku osi 1.4 program nasłuchujący SOAP korzysta z transakcji koordynowanych przez menedżera kolejek.

Klienty usługi WWW

Nadawcy SOAP nie są transakcyjni.

IBM MQ powiązania

Dla nadawcy SOAP można ustawić typ powiązania. Może on łączyć się jako aplikacja serwera IBM MQ lub jako aplikacja kliencka. Nadawcę SOAP można również powiązać jako klient XA w systemie .NET.

Trwałość komunikatu

Wybierz poziom trwałości, ustawiając opcję Persistence w identyfikatorze URI.

Transakcje usług WWW

Transakcji usług Web Service można używać, ponieważ nadawca SOAP nie jest transakcyjny. Jeśli użytkownik zapisuje własny nadawca SOAP i zamierza korzystać z transakcji usług Web Service, nie należy tworzyć transakcyjnego nadawcy SOAP. Nie można wysłać komunikatu żądania i odebrać komunikat odpowiedzi w tej samej transakcji. Wysyłanie i odbieranie nie może być koordynowane przez transakcję usługi Web Service.

Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service

Składnia i parametry w celu wdrożenia usługi Web Service produktu IBM MQ są definiowane w identyfikatorze URI. Program narzędziowy do wdrażania generuje domyślny identyfikator URI na podstawie nazwy usługi Web Service. Wartości domyślne można przestonić, definiując własny identyfikator URI jako parametr dla programu narzędziowego do wdrażania. Narzędzie wdrażające zawiera identyfikator URI w wygenerowanym kodzie pośredniczym klienta usługi Web Service.

Przeznaczenie

Usługa Web Service jest określana przy użyciu identyfikatora URI (Universal Resource Identifier). Diagram składni określa identyfikator URI, który jest obsługiwany w transporcie produktu IBM MQ dla protokołu SOAP. Identyfikator URI kontroluje specyficzne dla produktu IBM MQ parametry SOAP i opcje używane do uzyskiwania dostępu do usług docelowych. Identyfikator URI jest kompatybilny z usługami Web Services udostępnianym przez produkt .NET, Apache Axis 1, WebSphere Application Server, CICS.

Opis

Identyfikator URI jest włączany do klas klienta usługi Web Service wygenerowanych przez program narzędziowy wdrażania. Klient przekazuje identyfikator URI do nadawcy SOAP IBM MQ w komunikacie IBM MQ. Identyfikator URI steruje przetwarzaniem wykonywaną przez nadawcę SOAP IBM MQ SOAP Sender i IBM MQ SOAP.

Składnia

Składnia identyfikatora URI jest następująca:

```
jms:/queue?name=value&name=value...
```

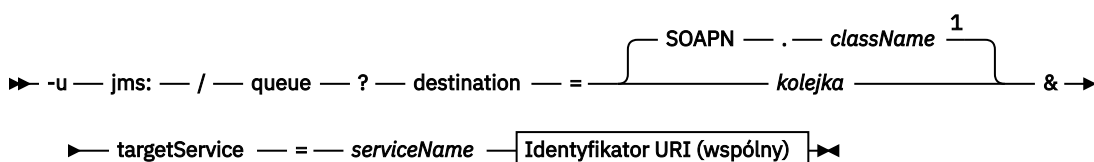
gdzie name jest nazwą parametru, a *wartość* jest odpowiednią wartością, a element name = *wartość* może być powtórzony dowolną liczbę razy z drugim, a kolejne wystąpienia poprzedzane ampersand (&).

W nazwach parametrów rozróżniana jest wielkość liter, podobnie jak nazwy obiektów IBM MQ. Jeśli dowolny parametr zostanie określony więcej niż jeden raz, końcowe wystąpienie parametru staje się skuteczne. Aplikacje klienckie mogą przestawiać wygenerowany parametr poprzez dodanie kolejnej kopii parametru do identyfikatora URI. Jeśli zostaną uwzględnione dodatkowe nierozpoznane parametry, zostaną one zignorowane.

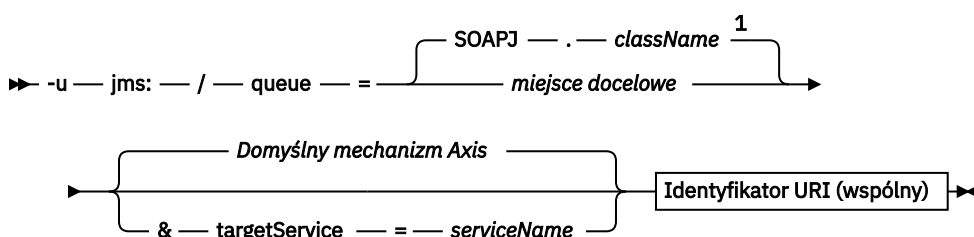
Jeśli identyfikator URI jest przechowywany w łańcuchu XML, należy reprezentować znak ampersand jako & ; . Podobnie, jeśli identyfikator URI jest zakodowany w skrypcie, należy zwrócić uwagę na znaki zmiany znaczenia, takie jak & , które w przeciwnym razie zostałyby zinterpretowane przez powłokę.

Diagram składni

Identyfikator URI (usługa.NET)

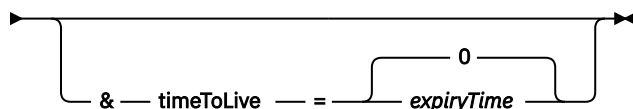
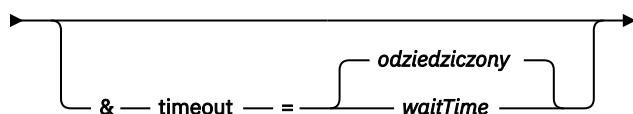
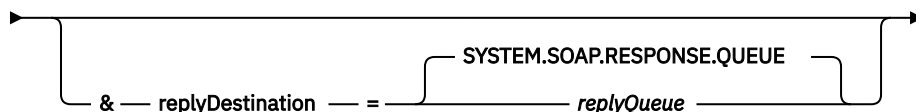
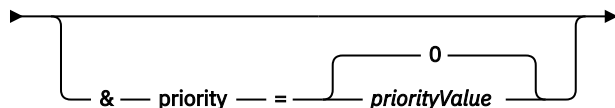
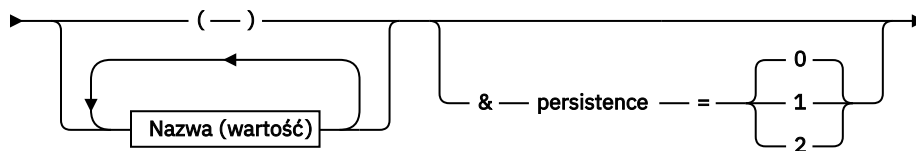


Identyfikator URI (usługa Java)

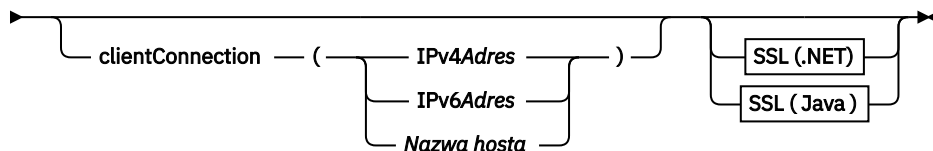
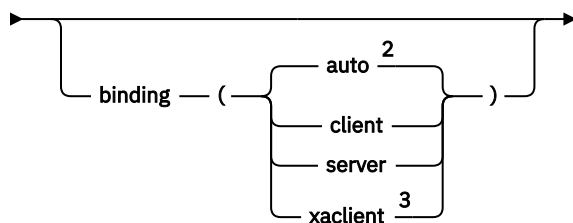
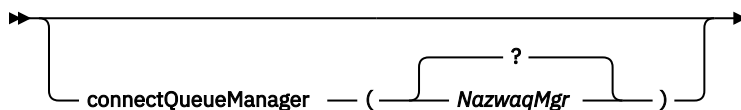


Identyfikator URI (wspólny)

➤ & — initialContextFactory — = — com.ibm.mq.jms.Nojndi — & — connectionFactory — = ➤



Nazwa (wartość)



Uwagi:

¹ Menedżer kolejek przekształca element *className* w nazwę kolejki zgodnie z krokami opisanymi w sekcji [“Transformacja nazwy miejsca docelowego na nazwę kolejki”](#) na stronie 1529 .

² *client* jest wartością domyślną, jeśli określono inne opcje odpowiednie dla klienta, na przykład *clientConnection* .

³ *xacient* ma zastosowanie tylko do .NET

Transformacja nazwy miejsca docelowego na nazwę kolejki

1. Parametr *className* jest poprzedzony przedrostkiem SOAPJ . dla usług Java lub z produktem SOAPN . dla usług .NET .
2. Rozszerzenie nazwy pliku jest usuwane z pełnej nazwy ścieżki podanej w parametrze *className* .
3. Wynikowy łańcuch jest obcinany do nie więcej niż 48 znaków
4. Znaki separatora katalogów są zastępowane znakami kropki.
5. Osadzone spacje są zastępowane znakami podkreślenia.
6. Dwukropek następujący po przedrostku napędu jest zastępowany kropką dla usługi .NET .

Uwaga: W niektórych środowiskach nazwa kolejki wygenerowana przez program narzędziowy wdrażania może nie być unikalna. Program narzędziowy do wdrażania sprawdza, czy ma zostać utworzona kolejka. Można przestonąć program narzędziowy wdrażania, restrukturyzując hierarchię katalogów wdrażania lub dostosowując dostarczony proces wdrażania.

Wymagane parametry identyfikatora URI

destination = kolejka

kolejka jest nazwą miejsca docelowego żądania. Może to być kolejka lub alias kolejki. Jeśli jest to alias kolejki, alias może zostać rozstrzygany do tematu.

- Jeśli parametr -u jest pominięty, *kolejka* jest generowana z klasy *nazwa_klasy* przy użyciu kroków opisanych w sekcji [“Transformacja nazwy miejsca docelowego na nazwę kolejki” na stronie 1529](#).
- Jeśli parametr -u jest określony jako *kolejka* , jest wymagany i musi być pierwszym parametrem identyfikatora URI po początkowym `jms: / queue ?` łańcuch. Określ nazwę kolejki produktu IBM MQ lub nazwę kolejki i nazwę menedżera kolejek połączoną z symbolem @, na przykład `SOAPN.trandemos@WMQSOAP.DEMO.QM`.
- Program narzędziowy do wdrażania sprawdza, czy nazwa kolejki, generowana lub podana, jest zgodna z nazwą istniejącej kolejki. Podjęte działanie jest opisane w sekcji [Tabela 187 na stronie 1530](#).

Tabela 187. Sprawdzanie poprawności kolejki			
Czy skrypt programu nastuchującego istnieje?	Skrypt programu nastuchującego istnieje w katalogu . /generated/server .		Skrypt nastuchiwania nie istnieje w katalogu . / generated/server
Kolejka w skrypcie nastuchiwania jest zgodna z kolejką ?	<i>queue</i> (kolejka) nie jest zgodna z kolejką żądań używaną w skrypcie nastuchiwania	<i>kolejka</i> jest zgodna z kolejką żądań używaną w skrypcie nastuchiwania	
kolejka istnieje	– Wdrażanie kończy się błędem. – Usługa została już wdrożona w produkcie . / generated/server, ale używana jest inna kolejka.	– Wdrożenie jest kontynuowane normalnie. – Usługa została już wdrożona w produkcie . / generated/server	– Wdrażanie kończy się błędem. – Skrypt uruchamiania programu nastuchującego nie został znaleziony w programie . / generated/server, ale <i>kolejka</i> jest używana przez inną usługę lub inną aplikację.
kolejka nie istnieje		– Wdrożenie jest kontynuowane z ostrzeżeniem. – Poprzednie wdrożenie mogło się nie powiodło, ponieważ uruchamianie jest poprawne, ale brakuje <i>kolejki</i> .	– Wdrożenie jest kontynuowane normalnie. – Z tego katalogu nie wdrożono żadnej usługi.

&connectionFactory = Nazwa (wartość)

Nazwa to jeden z następujących parametrów:

- [connectQueueManager \(qMgrNazwa\)](#)
- [binding \(bindingType\)](#)
- [clientChannel\(kanał\)](#)
- [clientConnection\(połączenie\)](#)
- [“Wymagane parametry SSL \(Java \)”](#) na stronie 1515

Opis wartości tych parametrów można znaleźć w sekcji [“Parametry fabryki połączeń”](#) na stronie 1532 .

&targetService = serviceName

⁹W systemie .NET *serviceName* jest nazwą usługi .NET , która znajduje się w katalogu wdrażania, na przykład: *targetService=myService.asmx*. W środowisku .NET parametr *targetService* umożliwia pojedynczy program nastuchujący SOAP produktu IBM MQ , który może przetwarzać żądania wielu usług. Te usługi muszą zostać wdrożone z tego samego katalogu.

⁹ Tylko usługa .NET

Opcjonalne parametry identyfikatora URI

&initialContextFactory = contextFactory

Parametr *contextFactory* jest wymagany i musi być ustawiony na wartość `com.ibm.mq.jms.NoJndi`. Upewnij się, że `NoJndi.jar` znajduje się w ścieżce klasy dla klienta usług Web Service produktu WebSphere Application Server. Program `NoJndi.jar` zwraca obiekty Java w oparciu o zawartość parametrów *connectionFactory* i *destination*, a nie przez odwołanie do katalogu.

&targetService = serviceName

¹⁰W przypadku osi *serviceName* jest to pełna nazwa usługi produktu Java, na przykład: `targetService=javaDemos.service.StockQuoteAxis`. Jeśli parametr *targetService* nie zostanie określony, usługa zostanie załadowana przy użyciu domyślnego mechanizmu Axis.

&persistence = messagePersistence

Parametr *messagePersistence* przyjmuje jedną z następujących wartości:

0

Trwałość jest dziedziczona z definicji kolejki.

1

Komunikat nie jest trwały.

2

Komunikat jest trwały

&priority = priorityValue

Wartość *priorityValue* mieści się w zakresie od 0 do 9. 0 oznacza niski priorytet. Wartość domyślna to specyficzna dla środowiska, która w przypadku IBM MQ ma wartość 0.

&replyDestination = KolejkareplyTo

Kolejka po stronie klienta, która ma być używana dla komunikatu odpowiedzi. Domyślną kolejką odpowiedzi jest `SYSTEM.SOAP.RESPONSE.QUEUE`.

- Uruchom skrypt `setupWMQSOAP`, aby utworzyć domyślne obiekty SOAP produktu IBM MQ.
- Określ kolejkę modelową dla kolejki *replyToQueue*, aby utworzyć tymczasową lub trwałą kolejkę odpowiedzi dynamicznej. W przypadku tymczasowych i trwałych kolejek dynamicznych odpowiedzi dla każdego żądania tworzona jest oddzielna instancja kolejki dynamicznej. Jeśli którekolwiek z poniższych zdarzeń zdarzy się, kolejka zostanie usunięta:
 - Odpowiedź dociera i jest przetwarzana.
 - Limit czasu żądania został wyczerpany.
 - Program żądający kończy działanie.

W celu uzyskania najlepszej wydajności należy używać tymczasowych kolejek dynamicznych, a nie stałych kolejek dynamicznych. Nie wysyłaj trwałego komunikatu żądania do identyfikatora URI z tymczasową kolejką dynamiczną. Proces nasłuchiwanie protokołu SOAP IBM MQ nie może przetworzyć komunikatu i wyjść z błędu. Limit czasu klienta oczekuje na odpowiedź.

- Skrypt `setupWMQSOAP` tworzy domyślną stałą dynamiczną kolejkę modelową o nazwie `SYSTEM.SOAP.MODEL.RESPONSE.QUEUE`.

&timeout = waitTime

Mierzony w milisekundach czas, przez jaki klient oczekuje na komunikat odpowiedzi. *waitTime* przestania wartości ustawione przez infrastrukturę lub aplikację kliencką. Jeśli nie zostanie określona wartość aplikacji, jeśli została określona, lub wartość domyślna infrastruktury jest dziedziczona.

Uwaga: Między limitem czasu i `timeToLive` nie jest wymuszana żadna relacja.

&timeToLive = expiryTime

expiryTime to czas określony w milisekundach przed utratą ważności komunikatu. Wartością domyślną jest zero, co oznacza nieograniczony czas życia.

Uwaga: Żaden związek nie jest wymuszany między limitem czasu i wartością `timeToLive`.

¹⁰ Tylko usługa Java

Parametry fabryki połączeń

connectQueueManager (*qMgrNazwa*)

Parametr *qMgrName* określa menedżer kolejek, z którym łączy się klient. Wartość domyślna jest pusta.

binding (*bindingType*)

bindingType określa, w jaki sposób klient jest połączony z nazwą *qMgrName*. Wartością domyślną jest auto. Element *bindingType* przyjmuje następujące wartości:

auto

Nadawca próbuje użyć następujących typów połączeń, w kolejności:

1. Jeśli zostaną określone inne opcje odpowiednie dla połączenia klienta, nadawca korzysta z powiązania klienta. Pozostałe opcje to clientConnection lub clientChannel.
2. Użyj połączenia z serwerem.
3. Użyj połączenia klienckiego.

Użyj opcji binding (auto) w polu *URI* , jeśli na kliencie SOAP nie ma lokalnego menedżera kolejek. Dla klienta SOAP zbudowano połączenie klienta.

client

Użyj opcji binding (klient) w polu *Identyfikator URI* , aby zbudować konfigurację klienta dla nadawcy SOAP.

server

Użyj opcji binding (serwer) w polu *Identyfikator URI* , aby zbudować konfigurację serwera dla nadawcy SOAP. Jeśli połączenie ma parametry typu klienta, połączenie nie powiedzie się i zostanie wyświetlony komunikat o błędzie przez nadawcę SOAP IBM MQ . Parametry typu klienta to clientConnection, clientChannel lub SSL.

xaclient

Parametr xaclient ma zastosowanie tylko w przypadku produktu .NET , a nie dla klientów Java . Użyj połączenia z klientem XA.

clientChannel (*kanal*)

Klient SOAP korzysta z *kanalu* w celu nawiązania połączenia z klientem IBM MQ . *kanal* musi być zgodny z nazwą kanału połączenia z serwerem, chyba że automatyczna definicja kanału jest włączona na serwerze. Parametr clientChannel jest wymaganym parametrem, o ile nie podano tabeli definicji połączeń klienta (CCDT).

Należy podać tabelę CCDT w produkcie Java , ustawiając wartość `com.ibm.mq.soap.transport.jms.mqchlurl`. W programie .NET ustaw zmienne środowiskowe MQCHLLIB i MQCHLTAB ; patrz "Użyj tabeli definicji kanału z transportem SOAP IBM MQ dla nadajnika SOAP" na stronie 1525.

clientConnection (*połączenie*)

Klient SOAP korzysta z *połączenia* w celu nawiązania połączenia z klientem IBM MQ . Domyślna nazwa hosta to localhost, a domyślny numer portu to 1414. Jeśli *połączenie* jest adresem TCP/IP, ma on jeden z trzech formatów i może być przyrostowy z numerem portu.

Klienty JMS mogą używać formatu : `hostname:port` lub 'escape' nawiasów kwadratowych, używając formatu `%X` , gdzie X jest wartością szesnastkową reprezentującą znak nawiasu na stronie kodowej identyfikatora URI. For example, in ASCII, %28 and %29 for (and) respectively.

Klienty .Net mogą używać nawiasów : `hostname(port)` w sposób jawny lub używać formatu "uszeregowanego".

IPv4 adres

Na przykład: 192.0.2.0.

IPv6 adres

Na przykład: 2001:DB8:0:0:0:0:0:0.

Nazwa hosta

Na przykład: `www.example.com%281687%29`, `www.example.com:1687` lub `www.example.com(1687)`.

SSL platforma

Więcej informacji znajduje się w sekcji [“Wymagane parametry SSL \(Java \)”](#) na stronie 1515

Przykładowe identyfikatory URI

Uwaga:

1. & w identyfikatorze URI jest kodowane jako & ;
2. Wszystkie wymienione wcześniej parametry mają zastosowanie do klientów.
3. Tylko produkty **destination**, **connectionFactory** i **initialContextFactory** mają zastosowanie do usługi WCF.

```
jms:/queue?  
destination=myQ&amp;connectionFactory=()&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Rysunek 23. Identyfikator URI usługi Axis, dostarczający tylko wymagane parametry

```
jms:/queue?destination=myQ&amp;connectionFactory=()&amp;targetService=MyService.asmx  
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Rysunek 24. Identyfikator URI usługi .NET , dostarczający tylko wymagane parametry

```
jms:/queue?destination=myQ@myRQM&amp;connectionFactory=connectQueueManager(myconnQM)  
binding(client)clientChannel(myChannel)clientConnection(myConnection)  
&amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Rysunek 25. Identyfikator URI usługi Axis, dostarczający niektórych opcjonalnych parametrów **connectionFactory**

```
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
```

Rysunek 26. Identyfikator URI usługi Axis, dostarczający opcji **sslPeerName** parametru **connectionFactory**.

Mechanizm Nojndi

Mechanizm Nojndi umożliwia programom JMS , które korzystają z interfejsów JNDI, korzystanie z tego samego identyfikatora URI co programy IBM MQ , które nie używają interfejsu JNDI.

Do wywoływania usług Web Service w produkcie WebSphere Application Server można użyć transportu IBM MQ dla protokołu SOAP. Produkt WebSphere Application Server SOAP over JMS wyszukuje zasoby produktu JMS przy użyciu interfejsu JNDI. Klient usługi Web Service może być uruchomiony na serwerze .NET lub za pomocą produktu Axis 1.4 w celu wywołania usługi Web Service, a nie za pomocą interfejsu JNDI. Aby użyć tego samego adresu URL dla klienta i serwera, należy podać te same informacje, czy środowisko korzysta z interfejsu JNDI, czy nie.

Identyfikator URI przekazany do transportu produktu IBM MQ dla protokołu SOAP przez klient usługi Web Service zawiera konkretne nazwy menedżerów kolejek i menedżerów kolejek produktu IBM MQ . Te nazwy są analizowane i używane bezpośrednio przez obsługę protokołu SOAP IBM MQ .

Mechanizm Nojndi kieruje fabryką **initialContext** używaną przez program JMS do produktu `com.ibm.mq.jms.Nojndi`. Klasa `com.ibm.mq.jms.Nojndi` to implementacja interfejsu JNDI, która zwraca obiekt **connectionFactory** i **destination** z adresu URL jako obiekty **ConnectionFactory** i **Queue** Java .

Jeśli implementacją produktu JMS są produkty IBM MQ, MQConnectionFactory i MQQueue dziedziczą z klas ConnectionFactory i Queue .

Za pomocą mechanizmu NoJndi można udostępnić te same informacje o połączeniu do serwera WebSphere Application Server i produktu .NET przy użyciu tego samego adresu URL.

Identyfikator URI W3C SOAP over JMS dla klienta IBM MQ Axis 2

Należy zdefiniować identyfikator URI W3C SOAP over JMS , aby wywołać usługę Web Service z klienta Axis 2 przy użyciu IBM MQ JMS jako transportu SOAP. Usługa Web Service musi być udostępniana przez serwer, który obsługuje IBM MQ JMS i W3C SOAP over JMS rekomendację kandydata dla powiązania SOAP/JMS .

Opis

Rekomendacja kandydatów W3C definiuje powiązanie protokołu SOAP korzystającego z usługi JMS (patrz: [Protokół SOAP korzystający z usługi Java Message Service 1.0](#)). Przydatny dla potrzeb przykładu jest również [Schemat identyfikatora URI dla usługi Java \(tm\) Message Service 1.0](#)¹¹.

Diagram składni służy do tworzenia identyfikatorów URI W3C SOAP over JMS , które są poprawne pod względem składniowym, i są akceptowane przez klienta IBM MQ Axis 2. Jest on ograniczony do zdefiniowania identyfikatora URI akceptowanego przez klienta IBM MQ Axis 2. Jest to podzbiór rekomendacji W3C w dwóch aspektach:

1. Parametr jms-variant topic nie jest obsługiwany i nie może być określony w identyfikatorze URI przekazanego do klienta produktu IBM MQ Axis 2.
2. Następujące właściwości są pomijane na diagramie składniowym, ponieważ są to właściwości produktu JMS , a nie część identyfikatora URI.
 - a. bindingVersion
 - b. contentType
 - c. soapAction
 - d. requestURI
 - e. isFault

Właściwości JMS są ustawiane przez klienta Axis 2 lub przez serwer.

Diagram rozszerza rekomendację W3C , definiując parametr niestandardowy connectionFactory. Element connectionFactory jest używany jako alternatywa dla interfejsu JNDI w celu określenia, w jaki sposób klient Axis 2 łączy się z menedżerem kolejek przy użyciu kolejki.

Klient IBM MQ Axis 2 akceptuje tylko właściwości jako część identyfikatora URI przekazanego do klienta przez aplikację kliencką lub jako zmienne środowiskowe. Klient IBM MQ Axis 2 nie ma możliwości przetwarzania dokumentu WSDL. Aplikacja kliencka lub narzędzie programistyczne może przetwarzać plik WSDL i utworzyć identyfikator URI do przekazania do klienta produktu Axis 2. Aplikacja kliencka IBM MQ w wersji 2 nie może bezpośrednio ustawiać właściwości komunikatu produktu JMS .

Składnia

Zgodnie z zaleceniem W3C wszystkie parametry mogą być uzyskiwane ze zmiennych środowiskowych. Nazwy zmiennych środowiskowych są tworzone z przedrostkiem nazwy parametru przy użyciu produktu soapjms_. Składnia jest następująca: soapjms_ *parameterName* ; na przykład,

```
set soapjms_targetServer=com.example.org.stockquote
```

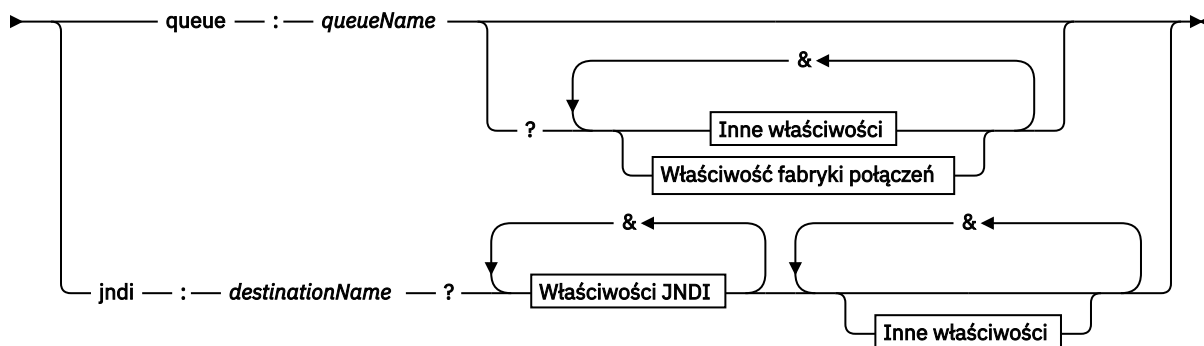
¹¹ W specyfikacji produktu W3C należy wyszukać *Schemat identyfikatora URI dla usługi JMS*, aby uzyskać najnowszą wersję roboczą.

Jeśli parametr jest ustawiony za pomocą zmiennej środowiskowej, nadpisuje ona wartość ustawioną w identyfikatorze URI.

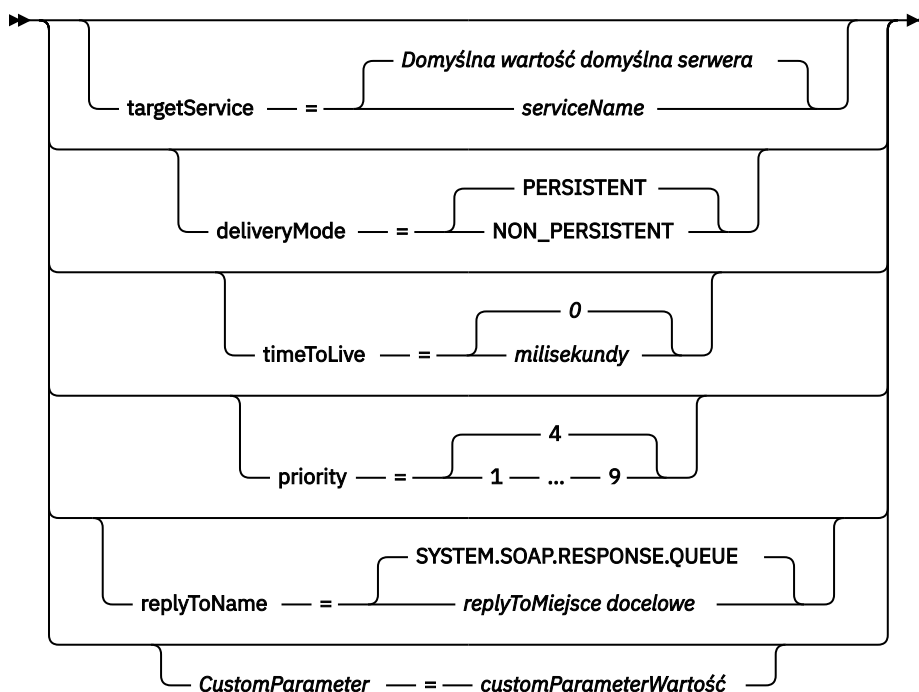
Zgodnie z zaleceniem W3C wszystkie parametry mogą być powtarzane. Używana jest ostatnia instancja parametru, o ile nie zostanie nadpisana przez zmienną środowiskową.

identyfikator_URI_JMS

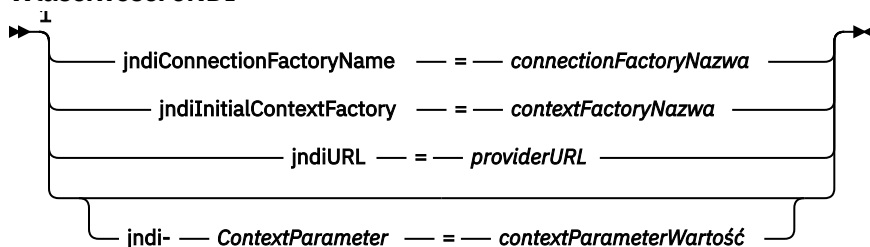
➔ jms: ➔



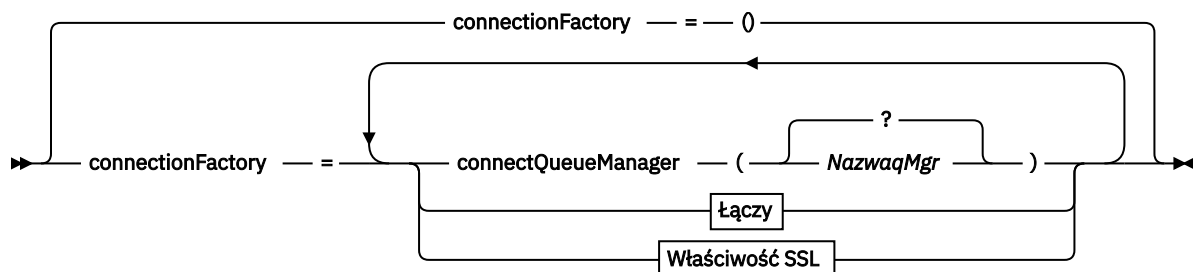
Inne właściwości



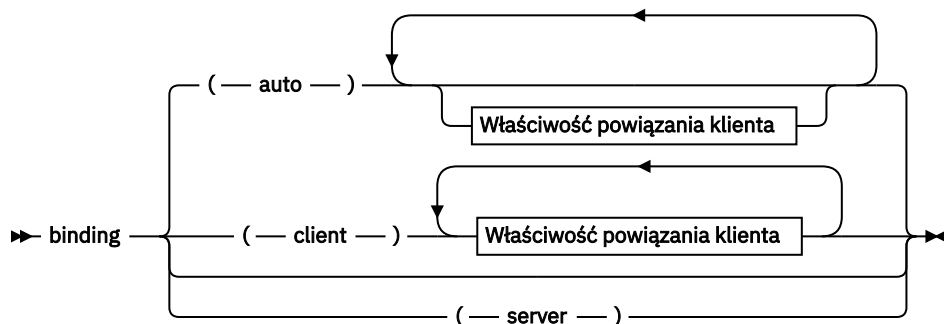
Właściwości JNDI



Właściwość fabryki połączeń

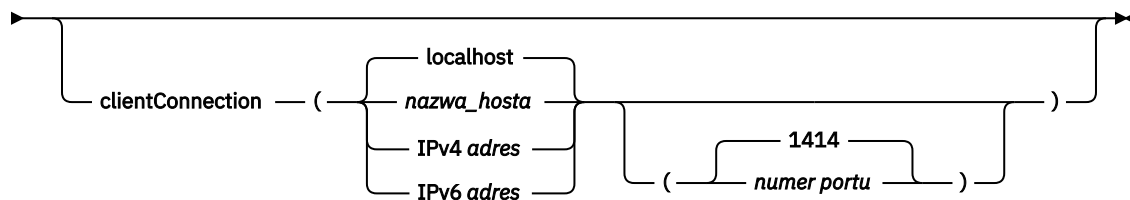


Łączy



Właściwość powiązania klienta

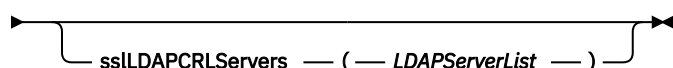
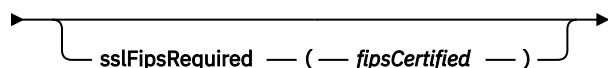
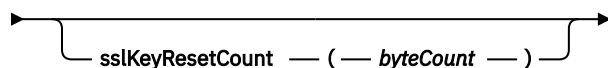
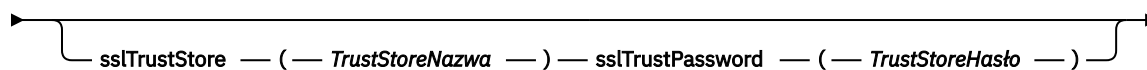
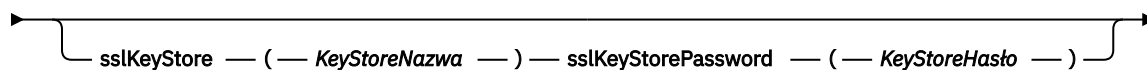
clientChannel — (— kanał —) →



Właściwość SSL

sslCipherSuite — (— cipherSuite —)

sslPeerName — (— peerName —)



Uwagi:

¹ Parametry `jndiConnectionFactoryName`, `jndiConnectionFactoryName` i `jndiURL` są wszystkimi wymaganymi parametrami. Opcja `jndi-ContextParameter` jest opcjonalna.

Parametry

connectionFactory = connectionFactoryParameterList

connectionFactoryParameterList to parametry, które kwalifikują się w sposób, w jaki klient Axis 2 łączy się z menedżerem kolejek, gdy wariant docelowy to queue.

connectionFactory nie może być określony z wariantem docelowym jndi .

Parametry nie są przekazywane do serwera w identyfikatorze URI żądania.

Jeśli parametr connectionFactory zostanie pominięty, kolejka musi należeć do domyślnego menedżera kolejek działającego na tym samym serwerze, co klient Axis 2.

Element *connectionFactoryParameterList*:

binding (bindingType)

bindingType określa, w jaki sposób klient jest połączony z nazwą *qMgrName*. Wartością domyślną jest auto. Element *bindingType* przyjmuje następujące wartości:

auto

Nadawca próbuje użyć następujących typów połączeń, w kolejności:

1. Jeśli zostaną określone inne opcje odpowiednie dla połączenia klienta, nadawca korzysta z powiązania klienta. Pozostałe opcje to clientConnection lub clientChannel.
2. Użyj połączenia z serwerem.
3. Użyj połączenia klienckiego.

Użyj opcji binding (auto) w polu *URI* , jeśli na kliencie SOAP nie ma lokalnego menedżera kolejek. Dla klienta SOAP zbudowano połączenie klienta.

client

Użyj opcji binding (klient) w polu *Identyfikator URI* , aby zbudować konfigurację klienta dla nadawcy SOAP.

server

Użyj opcji binding (serwer) w polu *Identyfikator URI* , aby zbudować konfigurację serwera dla nadawcy SOAP. Jeśli połączenie ma parametry typu klienta, połączenie nie powiedzie się i zostanie wyświetlony komunikat o błędzie przez nadawcę SOAP IBM MQ . Parametry typu klienta to clientConnection, clientChannel lub SSL.

xaclient

Parametr xaclient ma zastosowanie tylko w przypadku produktu .NET , a nie dla klientów Java . Użyj połączenia z klientem XA.

clientChannel (kanał)

Klient SOAP korzysta z *kanału* w celu nawiązania połączenia z klientem IBM MQ . *kanał* musi być zgodny z nazwą kanału połączenia z serwerem, chyba że automatyczna definicja kanału jest włączona na serwerze. Parametr clientChannel jest wymaganym parametrem, o ile nie podano tabeli definicji połączeń klienta (CCDT).

Należy podać tabelę CCDT w produkcie Java , ustawiając wartość `com.ibm.mq.soap.transport.jms.mqchlurl`. W programie .NET ustaw zmienne środowiskowe MQCHLLIB i MQCHLTAB ; patrz ["Użyj tabeli definicji kanału z transportem SOAP IBM MQ dla nadajnika SOAP"](#) na stronie 1525.

clientConnection (połączenie)

Klient SOAP korzysta z *połączenia* w celu nawiązania połączenia z klientem IBM MQ . Domyślna nazwa hosta to localhost, a domyślny numer portu to 1414. Jeśli *połączenie* jest adresem TCP/IP, ma on jeden z trzech formatów i może być przyrostowy z numerem portu.

Klienty JMS mogą używać formatu : `hostname:port` lub 'escape' nawiasów kwadratowych, używając formatu `%X` , gdzie X jest wartością szesnastkową reprezentującą znak nawiasu na stronie kodowej identyfikatora URI. For example, in ASCII, %28 and %29 for (and) respectively.

Klienty .Net mogą używać nawiasów : `hostname(port)` w sposób jawny lub używać formatu "uszeregowanego".

IPv4 adres

Na przykład: 192.0.2.0.

IPv6 adres

Na przykład: 2001:DB8:0:0:0:0:0:0.

Nazwa hosta

Na przykład: www.example.com%281687%29, www.example.com:1687lub
www.example.com(1687).

sslCipherSuite (*CipherSuite*)

CipherSuite określa parametr sslCipherSuite używany na kanale. Opcja CipherSuite określona przez klienta musi być zgodna z zestawem CipherSuite określonym w kanale połączenia serwera.

sslFipsRequired (*fipsCertified*)

fipsCertified określa, czy parametr *CipherSpec* lub *CipherSuite* musi używać kryptografii z certyfikatem FIPS w produkcie IBM MQ na kanale. Ustawienie parametru *fipsCertified* jest takie samo, jak ustawienie pola FipsRequired struktury **MQSCO** w wywołaniu MQCONN.

sslKeyStore (*KeyStoreNazwa*)

KeyStoreNazwa określa wartość sslKeyName używaną na kanale. Magazyn kluczy przechowuje klucz prywatny klienta używanego do uwierzytelniania klienta na serwerze. Magazyn kluczy jest opcjonalny, jeśli połączenie SSL zostało skonfigurowane w taki sposób, aby akceptować anonimowe połączenia klienta.

sslKeyResetCount (*bytecount*)

Parametr *bytecount* określa liczbę bajtów przekazywanych przez kanał SSL przed ponownym negocjacją klucza tajnego SSL. Aby wyłączyć renegotiację kluczy SSL, pominięte pole lub ustaw wartość zero. Wartość zero jest jedyną wartością obsługiwaną w niektórych środowiskach. Patrz sekcja [Renegotiowanie klucza tajnego w produkcie IBM MQ classes for Java](#). Efekt ustawienia sslKeyResetCount jest taki sam, jak ustawienie pola KeyResetCount w strukturze **MQSCO** w wywołaniu MQCONN.

sslKeyStorePassword (*KeyStoreHasło*)

KeyStoreHasło określa wartość sslKeyStorePassword używaną na kanale.

sslLDAPCRLServers (*LDAPServerList*)

LDAPServerList określa listę serwerów LDAP, które mają być używane na potrzeby sprawdzania listy odwołań certyfikatów.

W przypadku połączeń klienckich z obsługą SSL *LDAPServerList* to lista serwerów LDAP, które mają być używane do sprawdzania listy CRL (Certificate Revocation List). Certyfikat udostępniony przez menedżer kolejek jest sprawdzany w odniesieniu do jednego z wymienionych serwerów CRL LDAP. Jeśli zostanie znaleziony, połączenie nie powiedzie się. Każdy serwer LDAP jest podejmowany z kolei do czasu nawiązania połączenia z jednym z nich. Jeśli nawiązanie połączenia z żadnym z serwerów nie jest możliwe, certyfikat zostanie odrzucony. Po pomyślnym nawiązaniu połączenia z jednym z nich certyfikat jest akceptowany lub odrzucany w zależności od list CRL znajdujących się na tym serwerze LDAP.

Jeśli pole *LDAPServerList* jest puste, certyfikat należący do menedżera kolejek nie jest sprawdzany na liście odwołań certyfikatów. Jeśli podana lista identyfikatorów URI LDAP nie jest poprawna, zostanie wyświetlony komunikat o błędzie. Ustawienie tego pola jest takie samo, jak w przypadku rekordów MQAIR i uzyskiwanie dostępu do nich ze struktury **MQSCO** na serwerze MQCONN.

sslPeerName (*peerName*)

Parametr *peerName* określa nazwę sslPeerName używaną na kanale.

sslTrustStore (*TrustStoreNazwa*)

TrustStoreNazwa określa sslTrustStoreName używany na kanale. Magazyn zaufanych certyfikatów przechowuje publiczny certyfikat serwera lub jego łańcuch kluczy w celu uwierzytelnienia serwera na kliencie. Magazyn zaufanych certyfikatów jest opcjonalny, jeśli do uwierzytelniania serwera używany jest certyfikat główny ośrodka certyfikacji. W produkcie Javacertyfikaty główne są przechowywane w bazie certyfikatów środowiska JRE (cacerts).

sslTrustStorePassword (*TrustStoreHasło*)

TrustStoreHasło określa parametr sslTrustStorePassword używany na kanale.

CustomParameter = customParameterWartość

CustomParameter to nazwa zdefiniowana przez użytkownika dla parametru niestandardowego, a parametr *customParameter* jest wartością parametru.

Parametry niestandardowe, które nie są używane przez klienta Axis 2, są wysyłane przez klienta Axis 2 do serwera SOAP. Zapoznaj się z dokumentacją serwera. Parametr *connectionFactory* jest parametrem niestandardowym używanym przez klienta produktu Axis 2 i nie jest przekazywany do serwera.

Parametr *CustomParameter* nie może być zgodny z nazwą istniejącego parametru.

Jeśli parametr *CustomParameter* rozpoczyna się od łańcucha *jndi-* , jest on używany do wyszukiwania miejsca docelowego JNDI. Patrz sekcja [jndi-](#).

deliveryMode = deliveryMode

Parametr *deliveryMode* ustawia trwałość komunikatu. Wartością domyślną jest PERSISTENT.

jndi : destinationName

destinationName to nazwa miejsca docelowego JNDI, która jest odwzorowywać na kolejkę JMS . Jeśli określony jest wariant docelowy *jndi* , należy podać wartość *destinationName*.

jndiConnectionFactoryName = connectionFactoryNazwa

Klasa *connectionFactoryName* ustawia nazwę JNDI fabryki połączeń. Jeśli wariantem docelowym jest *jndi*, należy podać nazwę *connectionFactoryName* .

jndiInitialContextFactory = contextFactoryNazwa

Klasa *contextFactoryName* ustawia nazwę JNDI fabryki kontekstu początkowego. Jeśli wariantem docelowym jest *jndi*, należy podać wartość *contextFactoryName* . Więcej informacji na ten temat zawiera sekcja [Używanie interfejsu JNDI do pobierania obiektów administrowanych w aplikacji JMS](#).

jndiURL = providerURL

jndiURL ustawia nazwę adresu URL dostawcy JNDI. Jeśli wariantem docelowym jest *jndi*, należy podać wartość *jndiURL* .

jndi- ContextParameter = contextParameterWartość

jndi- ContextParameter jest nazwą zdefiniowaną przez użytkownika parametru niestandardowego, który służy do przekazywania informacji do dostawcy JNDI. *contextParameterWartość* to informacje, które są przekazywane.

priority = priorityValue

Parametr *priorityValue* ustawia priorytet komunikatu produktu JMS . 0 jest niski, 9 jest wysoki. Wartością domyślną jest 4.

queue : queueName

queueName to nazwa kolejki produktu JMS , w której umieszczane jest żądanie SOAP. Jeśli określono wariant kolejki, należy podać nazwę kolejki. Jeśli kolejka nie należy do domyślnego menedżera kolejek na tym samym serwerze co klient, należy ustawić parametr [connectionFactory](#) .

replyToName = replyToMiejsce docelowe

replyToMiejsce docelowe ustawia nazwę kolejki docelowej. Jeśli wariantem docelowym jest *jndi*, nazwa jest nazwą JNDI, która musi być odwzorowana na kolejkę. Jeśli wariantem jest kolejka , nazwa jest kolejką JMS . Wartością domyślną jest SYSTEM . SOAP . RESPONSE . QUEUE.

targetService = serviceName

Nazwa używana przez serwer SOAP do uruchamiania docelowej usługi Web Service.

W przypadku osi *serviceName* jest to pełna nazwa usługi produktu Java , na przykład: *targetService=www.example.org.StockQuote*. Jeśli parametr *targetService* nie zostanie określony, usługa zostanie załadowana przy użyciu domyślnego mechanizmu Axis.

timeToLive = milisekundy

Ustaw wartość *milisekundy* na czas, który upłynie przed utratą ważności komunikatu. Wartość domyślna 0 oznacza, że komunikat nigdy nie traci ważności.

Przykłady

```
jms:jndi:REQUESTQ
?jndiURL=file:/C:/JMSAdmin
&jndiInitialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory
&jndiConnectionFactoryName=ConnectionFactory
&replyToName=RESPONSEQ
&deliveryMode(NON_PERSISTENT)
```

Rysunek 27. Użyj komendy `jms:jndi`, aby wysłać żądanie SOAP/JMS

```
jms:queue:SOAPJ.demos
?connectionFactory=connectQueueManager(QM1)
Bind(Client)
ClientChannel(SOAPClient)
ClientConnection(www.example.org(1418))
&deliveryMode(NON_PERSISTENT)
```

Rysunek 28. Użyj komendy `jms: queue`, aby wysłać żądanie SOAP/JMS.

Obsługiwane usługi Web Service

Kod, który został napisany w celu uruchomienia jako usługi Web Service, nie musi być modyfikowany tak, aby używany był transport produktu IBM MQ dla protokołu SOAP. Należy wdrożyć usługi w inny sposób, aby były uruchamiane z transportem IBM MQ dla protokołu SOAP, a nie za pomocą protokołu HTTP.

Opis

V 8.0.0.2 W produkcie IBM MQ 8.0.0, Fix Pack 2 transport produktu IBM MQ dla protokołu SOAP jest nieaktualny. Więcej informacji na ten temat zawiera sekcja [Deprecations: IBM MQ transport SOAP client and HTTP Bridge](#).

Transport produktu IBM MQ dla protokołu SOAP udostępnia usługę nasłuchiwanie SOAP do uruchamiania usług dla środowiska .NET Framework 1 i .NET 2 oraz dla osi 1.4. Kanał niestandardowy produktu IBM MQ dla programu Microsoft Windows Communication Foundation uruchamia usługi dla produktu .NET Framework 3. Produkty WebSphere Application Server i CICS zapewniają obsługę uruchamiania usług za pośrednictwem protokołu SOAP w produkcie IBM MQ. Utwórz niestandardowy eksport w celu użycia produktu WebSphere Enterprise Service Bus lub WebSphere Process Server.

Program nasłuchujący SOAP produktu IBM MQ może przetwarzać żądania SOAP transakcyjnie. Uruchom program **amqwdployWMQService** przy użyciu opcji `-x`. Opcja dwufazowa jest obsługiwana tylko dla programów nasłuchujących korzystających z powiązań serwerów. Inne środowiska mogą udostępniać obsługę transakcyjną dla transportu produktu IBM MQ dla protokołu SOAP. Zapoznaj się z ich dokumentacją.

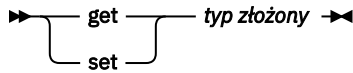
Transport produktu IBM MQ dla protokołu SOAP obecnie nie obsługuje standardowego protokołu SOAP over JMS, który został wprowadzony do usługi W3C. Istnieje możliwość odróżnienia komunikatu SOAP/JMS zapisanego do nowego standardu przez wyszukiwanie właściwości JMS BindingVersion. Transport IBM MQ dla protokołu SOAP nie ustawia właściwości BindingVersion.

Oś 1.4

Klasa Java może być zwykle używana bez modyfikacji. Typy dowolnych argumentów dla metod w usłudze Web Service muszą być obsługiwane przez mechanizm Axis. Szczegółowe informacje można znaleźć w dokumentacji produktu Axis. Jeśli usługa korzysta z obiektu złożonego jako argument lub zwraca jeden obiekt, ten obiekt musi być zgodny ze specyfikacją komponentu bean Java. Przykłady można znaleźć w następujących przykładach: [Rysunek 31 na stronie 1542](#), [Rysunek 32 na stronie 1543](#) i [Rysunek 33 na stronie 1543](#):

1. Mają publiczny konstruktor bezparametryczny.

2. Wszystkie typy złożone komponentu bean muszą mieć publiczne metody pobierające i ustawiające w postaci:



Przygotuj usługę do wdrożenia za pomocą programu narzędziowego **amqwdeployMQService** . Usługa jest wywoływana przez program nasłuchujący SOAP IBM MQ , który używa produktu axis.jar do uruchamiania usługi.

Jedynym dwufazowym menedżerem transakcji obsługiwany dla osi 1.4 jest IBM MQ.

Dostarczony program narzędziowy do wdrażania nie obsługuje sytuacji, w której usługa zwraca obiekt w innym pakiecie do samej usługi. Aby użyć obiektu zwróconego w innym pakiecie, należy napisać własny program narzędziowy do wdrażania. Korzystając z opcji -v , można oprzeć program narzędziowy wdrażania na dostarczonej próbce lub przechwytywać generowane przez niego komendy. Popraw komendy, aby utworzyć dostosowany skrypt.

Jeśli usługa korzysta z klas zewnętrznych w infrastrukturze Axis i środowisku wykonawczym SOAP produktu IBM MQ , należy ustawić poprawną wartość CLASSPATH. Aby zmienić CLASSPATH, należy zmienić wygenerowany skrypt, który uruchamia lub definiuje obiekty nasłuchiwanie w taki sposób, aby zawierały wymagane usługi, w jeden z następujących sposobów:

- Zmień CLASSPATH bezpośrednio w skrypcie po wywołaniu funkcji **amqwsetcp**.
- Utwórz skrypt specyficzny dla usługi, aby dostosować produkt CLASSPATH i wywołaj ten skrypt w wygenerowanym skrypcie po wywołaniu funkcji **amqwsetcp**.
- Utwórz dostosowany proces wdrażania, aby automatycznie dostosować produkt CLASSPATH w wygenerowanym skrypcie.

.NET Framework 1 i .NET Framework 2

Usługa, która została już przygotowana jako usługa Web Service HTTP, nie musi być modyfikowana w celu użycia jako usługa Web Service produktu IBM MQ . Musi zostać wdrożony za pomocą programu narzędziowego **amqwdeployMQService** .

Jedynym dwufazowym menedżerem transakcji obsługiwany dla .NET Framework 1 i .NET 2 jest produkt Microsoft Transaction Server (MTS).

Jeśli kod usługi nie został przygotowany jako usługa Web Service HTTP, należy przekształcić ją w usługę Web Service. Zadeklaruj klasę jako usługę Web Service i określ, w jaki sposób mają być formatowane parametry poszczególnych metod. Należy sprawdzić, czy wszystkie argumenty dotyczące metod usługi są zgodne ze środowiskiem. Rysunek 29 na stronie 1542 i Rysunek 30 na stronie 1542 przedstawiają klasę .NET , która została przygotowana jako usługa Web Service. Wprowadzone dodatki są oznaczone pogrubioną czcionką.

Produkt Rysunek 29 na stronie 1542 korzysta z modelu programowania za pomocą kodu w przypadku usługi Web Service produktu .NET . W modelu za pomocą kodu źródłowego źródło usługi jest oddzielone od pliku .asmx . Plik .asmx deklaruje nazwę powiązanego pliku źródłowego za pomocą słowa kluczowego Codebehind . Produkt IBM MQ zawiera przykłady zarówno wstawianych, jak i kodowych usług Web Services produktu .NET .

Źródło usług Web Service produktu .NET musi zostać skompilowane przed wdrożeniem przez program narzędziowy do wdrażania produktu **amqwdeployMQService** . Usługa jest kompilowana do biblioteki (.dll). Biblioteka musi być umieszczona w podkatalogu ./bin katalogu wdrażania.

.NET Framework 3

Utwórz niestandardowy kanał produktu IBM MQ dla programu Microsoft Windows Communication Foundation (WCF), aby wywołać usługi wdrożone w środowisku .NET 3. Opis sposobu konfigurowania produktu WCF w celu korzystania z transportu IBM MQ dla protokołu SOAP zawiera sekcja Tworzenie aplikacji WCF w produkcie IBM MQ .

WebSphere Application Server

Usługi Web Service udostępniane przez produkt WebSphere Application Server można wywoływać przy użyciu protokołu transportowego IBM MQ dla protokołu SOAP. Patrz [Korzystanie z protokołu SOAP przez produkt JMS do transportu usług Web Service](#).

Aby wygenerować klienta usług Web Service, należy zmodyfikować plik WSDL wygenerowany przez wdrożenie usługi JMS w produkcie WebSphere Application Server. Plik WSDL utworzony przez wdrożenie do produktu WebSphere Application Server zawiera identyfikator URI z odwołaniem JNDI do fabryki produktu JMS `InitialContext`. Należy zmodyfikować odwołanie JNDI do elementu `Nojndi` i udostępnić atrybuty połączenia zgodnie z opisem w sekcji ["Składnia i parametry identyfikatora URI dla wdrożenia usługi Web Service"](#) na stronie 1526.

CICS

Aplikacje produktu CICS można wywoływać przy użyciu protokołu transportowego IBM MQ dla protokołu SOAP. Patrz sekcja [Konfigurowanie systemu CICS dla usług Web Service](#).

WebSphere Enterprise Service Bus i WebSphere Process Server for Multiplatforms

Produkty WebSphere ESB i WebSphere Process Server for Multiplatforms obsługują protokół SOAP korzystający z produktu JMSz gotowym wbudowanym powiązaniem tylko wtedy, gdy używany jest domyślny dostawca przesyłania komunikatów produktu WebSphere Application Server. Utwórz niestandardowe powiązanie dla produktu JMS w celu obsługi transportu produktu IBM MQ dla protokołu SOAP. Patrz sekcja [Powiązania danych JMS](#).

Przykład

```
<%@ WebService Language="C#" CodeBehind="Quote.aspx.cs" Class="Quote.QuoteDotNet" %>
```

Rysunek 29. Definicja usługi dla .NET środowiska 2: *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();
        }
    }
}
```

Rysunek 30. Implementacja usługi dla .NET Framework 2: *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;
}
```

Rysunek 31. Interfejs usługi JAX-RPC produktu Java przy użyciu typu złożonego

```

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;
    }
}

```

Rysunek 32. Implementacja usługi JAX-RPC produktu Java przy użyciu typu złożonego

```

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; }
}

```

Rysunek 33. Implementacja komponentu bean usługi JAX-RPC Java bean o typie złożonym

Transport produktu IBM MQ dla klientów usług Web Service SOAP

Istnieje możliwość ponownego wykorzystania istniejącego klienta SOAP przez HTTP z transportem IBM MQ dla protokołu SOAP. Aby przekształcić klienta w pracę z transportem IBM MQ dla protokołu SOAP, należy wprowadzić pewne niewielkie modyfikacje w kodzie i procesie budowania.

Kod

Klienty JAX-RPC muszą być zapisywane w produkcie Java. Klienty .NET Framework 1 i 2 mogą być zapisywane w dowolnym języku, w którym używane jest środowisko wykonawcze Common Language Runtime. Przykłady kodu są udostępniane w języku C# i Visual Basic.

Poziom obsługi transakcji zależy od środowiska klienta i od wzorca interakcji SOAP. Żądanie SOAP i odpowiedź SOAP nie mogą być częścią tej samej transakcji atomowej.

Produkt IBM.WMQSOAP.Register.Extension() należy wywołać w kliencie środowiska .NET Framework 1, .NET Framework 2. W kliencie usługi Web Service JAX-RPC Java wywołaj `com.ibm.mq.soap.Register.extension`, aby zarejestrować nadawcę SOAP produktu IBM MQ. Metoda rejestruje transport produktu IBM MQ dla nadawcy SOAP jako procedurę obsługi dla komunikatów SOAP przy użyciu protokołu jms: .

Aby utworzyć klient .NET Framework 3, wygeneruj proxy klienta programu Windows Communication Foundation przy użyciu narzędzia **svcutil**. Patrz sekcja [Generowanie plików proxy klienta WCF i plików konfiguracyjnych aplikacji przy użyciu narzędzia svcutil z metadanymi z działającej usługi](#).

Biblioteki wymagane do budowania i uruchamiania klientów .NET Framework 1 i 2

- amqsoap
- System
- System.Web.Services
- System.Xml

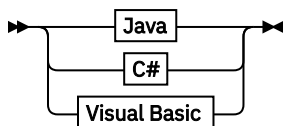
Biblioteki wymagane do budowania i uruchamiania klientów Axis 1.4

- `MQ_Install\java\lib\com.ibm.mq.soap.jar`;

- `MQ_Install\java\lib\com.ibm.mq.commonservices.jar;`
- `MQ_Install\java\lib\soap\axis.jar;`
- `MQ_Install\java\lib\soap\jaxrpc.jar`
- `MQ_Install\java\lib\soap\saaaj.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;`

Uwaga: Produkty IBM MQ 8.0, `ldap.jar`, `jndi.jar` i `jta.jar` zostały usunięte z tej listy, ponieważ są one obecnie częścią pakietu JDK.

Zarejestruj rozszerzenie SOAP



Java

➡ `com.ibm.mq.soap.Register.extension()` ➡

C#

➡ `IBM.WMQSOAP.Register.Extension();` ➡

Visual Basic

➡ `IBM.WMQSOAP.Register.Extension` ➡

Przykłady klientów

Rysunek 34 na stronie 1544 to przykład klienta C# środowiska .NET Framework 1 lub .NET Framework 2, który korzysta z modelu programowania wstawianego. Metoda **IBM.WMQSOAP.Register.Extension()** rejestruje nadawcę SOAP IBM MQ przy użyciu programu .NET jako procedury obsługi protokołu jms: .

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
            } catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
            }
        }
    }
}
```

Rysunek 34. C# Przykład klienta usługi Web Service

Rysunek 35 na stronie 1545 to przykład klienta produktu Java, który korzysta z statycznego interfejsu klienta proxy JAX-RPC. Metoda **com.ibm.mq.soap.Register.extension()**; rejestruje nadawcę SOAP IBM MQ przy użyciu proxy usługi w celu obsługi protokołu jms: .

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
    public static void main(String[] args) {
        try {
            Register.extension();
            QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
            System.out.println("Response = "
                + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
        } catch (Exception e) {
            System.out.println("Exception = " + e.getMessage());
        }
    }
}
```

Rysunek 35. Przykład klienta usługi Web Service produktu Java

Procedury zewnętrzne, wyjścia funkcji API i odwołania do usług instalowalnych

Odsyłacze zawarte w tej sekcji ułatwiają programowanie wyjść użytkownika, wyjść funkcji API i aplikacji usług instalowalnych:

- [“Struktura MQIEP” na stronie 1546](#)
- [“Dane wyjściowe wyjścia konwersji danych” na stronie 1549](#)
- [“MQ_PUBLISH_EXIT-wyjście publikowania” na stronie 1553](#)
- [“Wywołania wyjścia kanału i struktury danych” na stronie 1561](#)
- [“Odwołanie do wyjścia funkcji API” na stronie 1652](#)
- [“Informacje uzupełniające o interfejsie usług instalowalnych” na stronie 1714](#)
-  [“Informacje uzupełniające o interfejsie usług instalowalnych” na stronie 1779](#)

Pojęcia pokrewne

[“Skorowidz aplikacji MQI” na stronie 7](#)

Odsyłacze zawarte w tej sekcji ułatwiają programowanie aplikacji interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Odsyłacze pokrewne

[“SOAP-odwołanie” na stronie 1486](#)

Transport IBM MQ dla informacji o odwołaniu SOAP ułożonych alfabetycznie.

[“Materiał odniesienia dla IBM MQ bridge for HTTP” na stronie 1819](#)

Tematy dotyczące produktu IBM MQ bridge for HTTP, uporządkowane alfabetycznie

[“Klasy i interfejsy IBM MQ .NET” na stronie 1856](#)

Klasy i interfejsy IBM MQ .NET są uporządkowane alfabetycznie. Opisywane są właściwości, metody i konstruktory.

[“Klasy C++ w programie IBM MQ” na stronie 1921](#)

Klasy języka C++ programu IBM MQ hermetyzują interfejs kolejki komunikatów produktu IBM MQ (MQI). Dostępny jest pojedynczy plik nagłówkowy C++ **imqi.hpp**, który obejmuje wszystkie te klasy.

Informacje pokrewne

[Projektowanie aplikacji](#)

[Klasy IBM MQ dla bibliotek produktu Java](#)

[Klasy IBM MQ dla JMS](#)

Struktura MQIEP

Struktura MQIEP zawiera punkt wejścia dla każdego wywołania funkcji, które jest dozwolone w celu wykonania wyjść.

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQIEP_STRUC_ID

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQIEP_VERSION_1

Numer wersji struktury wersji 1.

MQIEP_CURRENT_VERSION

Bieżąca wersja struktury.

StrucLength

Typ: MQLONG

Wielkość struktury MQIEP w bajtach. Wartość jest następująca:

MQIEP_LENGTH_1

Flagi

Typ: MQLONG

Zawiera informacje na temat adresów funkcji. Flaga wskazująca, czy biblioteka jest wielowątkowa, może być używana z flagą w celu wskazania, czy biblioteka jest biblioteką klienta, czy serwera.

Do określenia informacji o bibliotece używana jest następująca wartość:

MQIEPF_NONE

Jedna z następujących wartości jest używana do określenia, czy biblioteka współużytkowana jest wielowątkowa, czy też nie:

BIBLIOTEKA MQIEPF_NON_THREADED_LIBRARY

Niewątkowa biblioteka współużytkowana

MQIEPF_THREADED_LIBRARY

Wielowątkowa biblioteka współużytkowana

Jedna z następujących wartości jest używana do określenia, czy biblioteka współużytkowana jest biblioteką współużytkowaną klienta, czy też serwerem:

MQIEPF_CLIENT_LIBRARY

Biblioteka współużytkowana klienta

MQIEPF_LOCAL_LIBRARY

Biblioteka współużytkowana serwera

Zarezerwowane

Typ: MQPTR

Wywołanie MQBACK_Call

Typ: PMQ_BACK_CALL

Adres wywołania MQBACK.

Wywołanie MQBEGIN_Call

Typ: PMQ_BEGIN_CALL

Adres wywołania MQBEGIN.

Wywołanie MQBUFMH_Call

Typ: PMQ_BUFMH_CALL

Adres wywołania MQBUFMH.

Wywołania MQCB_Call

Typ: PMQ_CB_CALL

Adres wywołania MQCB.

Wywołanie MQCLOSE_Call

Typ: PMQ_CLOSE_CALL

Adres wywołania MQCLOSE.

Wywołanie MQCMIT_Call

Typ: PMQ_CMIT_CALL

Adres wywołania MQCMIT.

Wywołanie MQCONN_Call

Typ: PMQ_CONN_CALL

Adres wywołania MQCONN.

Wywołanie MQCONNX_Call

Typ: PMQ_CONNX_CALL

Adres wywołania MQCONNX.

Wywołanie MQCRTMH_Call

Typ: PMQ_CRTMH_CALL

Adres wywołania MQCRTMH.

Wywołanie MQCTL_Call

Typ: PMQ_CTL_CALL

Adres wywołania MQCTL.

Wywołanie MQDISC_Call

Typ: PMQ_DISC_CALL

Adres wywołania MQDISC.

Wywołanie MQDLTMH_Call

Typ: PMQ_DLTMH_CALL

Adres wywołania MQDLTMH.

Wywołanie MQDLTMP_Call

Typ: PMQ_DLTMP_CALL

Adres wywołania MQDLTMP.

Wywołanie MQGET_Call

Typ: PMQ_GET_CALL

Adres wywołania MQGET.

Wywołanie MQINQ_Call

Typ: PMQ_INQ_CALL

Adres wywołania MQINQ.

Wywołanie MQINQMP_Call

Typ: PMQ_INQMP_CALL

Adres wywołania MQINQMP.

MQMHBUF_Call

Typ: PMQ_MHBUF_CALL

Adres wywołania MQMHBUF.

Wywołanie MQOPEN_Call

Typ: PMQ_OPEN_CALL

Adres wywołania MQOPEN.

Wywołanie MQPUT_Call

Typ: PMQ_PUT_CALL

Adres wywołania MQPUT.

MQPUT1_Call

Typ: PMQ_PUT1_CALL

Adres wywołania MQPUT1 .

Wywołanie MQSET_Call

Typ: PMQ_SET_CALL

Adres wywołania MQSET.

Wywołanie MQSETMP_Call

Typ: PMQ_SETMP_CALL

Adres wywołania MQSETMP.

Wywołanie MQSTAT_Call

Typ: PMQ_STAT_CALL

Adres wywołania MQSTAT.

Wywołanie MQSUB_Call

Typ: PMQ_SUB_CALL

Adres wywołania MQSUB.

Wywołanie MQSUBRQ_Call

Typ: PMQ_SUBRQ_CALL

Adres wywołania MQSUBRQ.

Wywołanie MQXCNVC_Call

Typ: PMQ_XCNVC_CALL

Adres wywołania MQXCNVC.

Wywołanie MQXCLWLN_Call

Typ: PMQ_XCLWLN_CALL

Adres wywołania MQXCLWLN.

Wywołanie MQXDX_Call

Typ: PMQ_XDX_CALL

Adres wywołania MQXDX.

Wywołanie MQXEP_Call

Typ: PMQ_XEP_CALL

Adres wywołania MQXEP.

Wywołanie MQZEP_Call

Typ: PMQ_ZEP_CALL

Adres wywołania MQZEP.

Deklaracja C

```
struct tagMQIEP {
    MQCHAR4      StrucId;          /* Structure identifier */
    MQLONG        Version;         /* Structure version number */
    MQLONG        StrucLength;     /* Structure length */
    MQLONG        Flags;           /* Flags */
    MQPTR         Reserved;        /* Reserved */
    PMQ_BACK_CALL MQBACK_Call;     /* Address of MQBACK */
    PMQ_BEGIN_CALL MQBEGIN_Call;   /* Address of MQBEGIN */
    PMQ_BUFMH_CALL MQBUFMH_Call;   /* Address of MQBUFMH */
    PMQ_CB_CALL   MQCB_Call;       /* Address of MQCB */
    PMQ_CLOSE_CALL MQCLOSE_Call;   /* Address of MQCLOSE */
    PMQ_CMIT_CALL MQCMIT_Call;     /* Address of MQCMIT */
    PMQ_CONN_CALL MQCONN_Call;     /* Address of MQCONN */
    PMQ_CONNX_CALL MQCONNX_Call;   /* Address of MQCONNX */
    PMQ_CRTMH_CALL MQCRTMH_Call;   /* Address of MQCRTMH */
    PMQ_CTL_CALL   MQCTL_Call;     /* Address of MQCTL */
    PMQ_DISC_CALL  MQDISC_Call;    /* Address of MQDISC */
    PMQ_DLTMH_CALL MQDLTMH_Call;   /* Address of MQDLTMH */
    PMQ_DLTMP_CALL MQDLTMP_Call;   /* Address of MQDLTMP */
    PMQ_GET_CALL   MQGET_Call;     /* Address of MQGET */
    PMQ_INQ_CALL   MQINQ_Call;     /* Address of MQINQ */
    PMQ_INQMP_CALL MQINQMP_Call;   /* Address of MQINQMP */
    PMQ_MHBUF_CALL MQMHBUF_Call;   /* Address of MQMHBUF */
    PMQ_OPEN_CALL  MQOPEN_Call;    /* Address of MQOPEN */
    PMQ_PUT_CALL   MQPUT_Call;     /* Address of MQPUT */
    PMQ_PUT1_CALL  MQPUT1_Call;    /* Address of MQPUT1 */
    PMQ_SET_CALL   MQSET_Call;     /* Address of MQSET */
    PMQ_SETMP_CALL MQSETMP_Call;   /* Address of MQSETMP */
    PMQ_STAT_CALL  MQSTAT_Call;    /* Address of MQSTAT */
    PMQ_SUB_CALL   MQSUB_Call;     /* Address of MQSUB */
    PMQ_SUBRQ_CALL MQSUBRQ_Call;   /* Address of MQSUBRQ */
    PMQ_XCLWLN_CALL MQXCLWLN_Call; /* Address of MQXCLWLN */
    PMQ_XCNVC_CALL MQXCNVC_Call;   /* Address of MQXCNVC */
    PMQ_XDX_CALL   MQXDX_Call;     /* Address of MQXDX */
    PMQ_XEP_CALL   MQXEP_Call;     /* Address of MQXEP */
    PMQ_ZEP_CALL   MQZEP_Call;     /* Address of MQZEP */
};
```

Dane wyjściowe wyjścia konwersji danych

W przypadku produktu z/OS konieczne jest zapisanie wyjść konwersji danych w języku assembler. W przypadku innych platform zaleca się korzystanie z języka programowania C.

Aby ułatwić utworzenie programu obsługi wyjścia konwersji danych, dostarczane są następujące informacje:

- Plik źródłowy szkieletu
- Wywołanie konwersji znaków
- Program narzędziowy, który tworzy fragment kodu, który wykonuje konwersję danych w strukturach typu danych. Ten program narzędziowy korzysta tylko z danych wejściowych w języku C. W systemie z/OS stworzony jest kod assemblera.

Aby procedura pisania programów była widoczna:

-  [Pisanie programu obsługi wyjścia konwersji danych dla programu IBM MQ for IBM i](#)
-  [Pisanie programu obsługi wyjścia konwersji danych dla programu IBM MQ for z/OS](#)
- [Pisanie wyjścia konwersji danych dla systemu IBM MQ w systemach UNIX i Linux](#)
- [Pisanie wyjścia konwersji danych dla programu IBM MQ for Windows](#)

Plik źródłowy szkieletu

Mogą one być używane jako punkt wyjścia podczas pisania programu obsługi wyjścia konwersji danych.

Podane pliki są wymienione w sekcji [Tabela 188](#) na stronie 1550.

Tabela 188. Pliki źródłowe szkieletu

Platforma	Plik
AIX	amqsvfc0.c
IBM i	QMQMSAMP/QCSRC (AMQSVFC4)
HP-UX	amqsvfc0.c
Linux	amqsvfc0.c
z/OS	CSQ4BAX8 (“1” na stronie 1550) CSQ4BAX9 (“2” na stronie 1550) CSQ4CAX9 (“3” na stronie 1550)
Solaris	amqsvfc0.c
Systemy Windows	amqsvfc0.c
Uwagi: 1. Ilustruje wywołanie MQXCVNC. 2. Opakowanie dla fragmentów kodu wygenerowanych przez program narzędziowy do użycia we wszystkich środowiskach z wyjątkiem CICS. 3. Opakowanie dla fragmentów kodu wygenerowanych przez program narzędziowy do użycia w środowisku CICS .	

Przekształć znaki w wywołaniu

Należy użyć wywołania MQXCNVC (przekształć znaki) z programu obsługi wyjścia konwersji danych w celu przekształcenia danych komunikatu znakowego z jednego zestawu znaków na inny. W przypadku niektórych wielobajtowych zestawów znaków (na przykład zestawów znaków UCS-2) należy użyć odpowiednich opcji.

Żadne inne wywołania MQI nie mogą być wykonywane z poziomu wyjścia. Próba wykonania takiego wywołania nie powiodła się. Kod przyczyny: MQRC_CALL_IN_PROGRESS.

Więcej informacji na temat wywołania MQXCNVC i odpowiednich opcji zawiera sekcja [“MQXCNVC-Przekształć znaki” na stronie 931](#) .

Program narzędziowy do tworzenia kodu wyjścia konwersji

Te informacje umożliwiają zapoznanie się z informacjami na temat tworzenia kodu wyjścia konwersji.

Komendy służące do tworzenia kodu wyjścia konwersji są następujące:

IBM i

CVTMQMDTA (Konwersja Typu Danych IBM MQ)

Windows, UNIX i Linux

crtmqcvx (Tworzenie konwersji IBM MQ -wyjście)

 **z/OS**
CSQUCVX

Komenda dla używanej platformy tworzy fragment kodu, który wykonuje konwersję danych w strukturach typu danych, do użycia w programie obsługi wyjścia konwersji danych. Komenda pobiera plik zawierający jedną lub więcej definicji struktury języka C.  W systemie z/OS następnie generowany jest zestaw danych zawierający fragmenty kodu asemblera i funkcje konwersji. Na innych platformach generuje on plik z funkcją C, aby przekształcić każdą definicję struktury. W systemie z/OS program narzędziowy wymaga dostępu do biblioteki wykonawczej SCEERUN LE/370.

Wywoływanie programu narzędziowego CSQUCVX w systemie z/OS



Rysunek 36 na stronie 1551 przedstawia przykład kodu JCL użytego do wywołania programu narzędziowego CSQUCVX.

```
//CVX      EXEC PGM=CSQUCVX
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
//          DD DISP=SHR,DSN=thlqual.SCSQLOAD
//          DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP  DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT  DD DISP=OLD,DSN=MY.MQSERIES.EXIT(SMSG1)
```

Rysunek 36. Przykładowy skrypt JCL używany do wywoływania programu narzędziowego CSQUCVX

Instrukcje definicji danych produktu z/OS



Program narzędziowy CSQUCVX wymaga instrukcji DD z następującymi nazwami DDnames:

Instrukcja DD	opis
SYSPRINT	Określa klasę zestawu danych lub klasę buforowania wydruku dla raportów i komunikatów o błędach.
CSQUINP	Określa partycjonowany zestaw danych zawierający definicje struktur danych, które mają zostać przekształcone.
CSQUOUT	Określa partycjonowany zestaw danych, w którym mają zostać zapisane fragmenty kodu konwersji. Długość rekordu logicznego (LRECL) musi wynosić 80, a format rekordu (RECFM) musi mieć wartość FB.

Komunikaty o błędach w systemach Windowsi UNIX i Linux

Komenda cstrmqcvx zwraca komunikaty z zakresu od AMQ7953 do AMQ7970.

Komunikaty te są wyświetlane w programie [Komunikaty i kody przyczyny IBM MQ Komunikaty](#).

Istnieją dwa główne typy błędów:

- Poważne błędy, takie jak błędy składniowe, gdy przetwarzanie nie może być kontynuowane.

Na ekranie wyświetlany jest komunikat zawierający numer wiersza błędu w pliku wejściowym. Możliwe, że plik wyjściowy został częściowo utworzony.

- Inne błędy, gdy wyświetlany jest komunikat informujący o tym, że wystąpił problem, ale analizowanie struktury może być kontynuowane.

Plik wyjściowy został utworzony i zawiera informacje o błędach, jakie wystąpiły. Ta informacja o błędzie jest poprzedzona przedrostkiem #error, dzięki czemu wygenerowany kod nie jest akceptowany przez żaden kompilator bez interwencji w celu naprawienia problemów.

Poprawna składnia

Plik wejściowy dla programu narzędziowego musi być zgodny ze składnią języka C.

Jeśli użytkownik nie zna języka C, należy zapoznać się z tematem [Przykład w języku C](#) w tym temacie.

Ponadto należy pamiętać o następujących regułach:

- typedef jest rozpoznawany tylko przed słowem kluczowym struct.
- W deklaracjach struktury wymagany jest znacznik struktury.
- Aby oznaczyć tablicę o zmiennej długości lub łańcuch na końcu komunikatu, można użyć pustych nawiasów kwadratowych [].
- Tablice wielowymiarowe i tablice łańcuchów nie są obsługiwane.
- Rozpoznawane są następujące dodatkowe typy danych:
 - MQBOOL
 - MQBYTE
 - MQCHAR
 - MQFLOAT32
 - MQFLOAT64
 - MQSHORT
 - MQLONG
 - MQINT8
 - MQUINT8
 - MQINT16
 - MQUINT16
 - MQINT32
 - MQUINT32
 - MQINT64
 - MQUINT64

Pola MQCHAR są przekształcane na stronę kodową, ale tabele MQBYTE, MQINT8 i MQUINT8 są pozostawiane bez zmian. Jeśli kodowanie jest inne, MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 i MQBOOL są odpowiednio przekształcane.

- Nie należy używać następujących typów danych:
 - double (podwójna)
 - wskaźniki
 - bit-fields

Jest to spowodowane tym, że program narzędziowy do tworzenia kodu wyjścia konwersji nie udostępnia narzędzia do konwersji tych typów danych. Aby to przezwyciężyć, możesz napisać swoje własne podprogramy i zadzwonić do nich z wyjścia.

Inne punkty do nota:

- Nie należy używać numerów kolejnych w wejściowym zestawie danych.
- Jeśli istnieją pola, dla których mają zostać utworzone własne procedury konwersji, należy je zadeklarować jako wartość MQBYTE, a następnie zastąpić wygenerowane makra CMQXCFBA własnym kodem konwersji.

Przykład C

```
struct TEST { MQLONG    SERIAL_NUMBER;  
              MQCHAR     ID[5];  
              MQINT16     VERSION;  
              MQBYTE      CODE[4];  
              MQLONG      DIMENSIONS[3];  
              MQCHAR      NAME[24];  
            } ;
```

Odnosi się to do następujących deklaracji w innych językach programowania:

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

Obsługiwane tylko w systemie z/OS

```
DCL 1 TEST,  
  2 SERIAL_NUMBER FIXED BIN(31),  
  2 ID             CHAR(5),  
  2 VERSION        FIXED BIN(15),  
  2 CODE           CHAR(4),          /* not to be converted */  
  2 DIMENSIONS(3)  FIXED BIN(31),  
  2 NAME           CHAR(24);
```

MQ_PUBLISH_EXIT-wyjście publikowania

Wywołanie MQ_PUBLISH_EXIT może sprawdzać i zmieniać komunikaty dostarczane do subskrybentów.

Przeznaczenie

Wyjście publikowania służy do sprawdzania i modyfikowania komunikatów dostarczanych do subskrybentów:

- Sprawdzanie treści komunikatu publikowanego dla każdego subskrybenta
- Modyfikowanie treści komunikatu publikowanego dla każdego subskrybenta
- Zmien kolejkę, do której jest wstawiany komunikat
- Zatrzymaj dostarczanie komunikatu do subskrybenta

To wyjście nie jest dostępne w produkcie IBM MQ for z/OS.

Składnia

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

Parametry

ExitParms (MQPSXP) - Input/Output

ExitParms zawiera informacje na temat wywołania wyjścia.

PubContext (MQPBC) - Input

PubContext zawiera informacje kontekstowe dotyczące publikatora publikacji.

SubContext (MQSBC) - Input/Output

SubContext zawiera kontekstowe informacje o subskrybencie otrzymującego publikację.

MQPSXP-Publikowanie struktury danych wyjścia

Struktura MQPSXP opisuje informacje, które są przekazywane do wyjścia publikowania i zwracane z niego.

Tabela 189 na stronie 1554 podsumowuje pola w strukturze:

Tabela 189. Pola w MQPSXP	
Pole	Opis
<u>StrucID</u>	Identyfikator struktury
<u>Version</u>	Numer wersji struktury
<u>ExitId</u>	Typ wywołanej procedury zewnętrznej
<u>ExitReason</u>	Przyczyna wywołania wyjścia
<u>ExitResponse</u>	Odpowiedź od wyjścia
<u>ExitResponse2</u>	Odpowiedź dodatkowa z wyjścia
<u>Feedback</u>	Kod zwrotny
<u>ExitUserArea</u>	Wyjdź z obszaru użytkownika
<u>ExitData</u>	Dane wyjścia
<u>QMgrName</u>	Nazwa lokalnego menedżera kolejek
<u>Hconn</u>	Uchwyt połączenia
<u>MsgDescPtr</u>	Adres deskryptora komunikatu (MQMD)
<u>MsgHandle</u>	Uchwyt do właściwości komunikatu (MQHMSG)
<u>MsgInPtr</u>	Adres komunikatu wejściowego
<u>MsgInLength</u>	Długość komunikatu wejściowego
<u>MsgOutPtr</u>	Adres komunikatu wyjściowego
<u>MsgOutLength</u>	Długość komunikatu wyjściowego
<u>pEntryPoints</u>	Adres struktury MQIEP

Pola

StrucID (MQCHAR4)

StrucID jest identyfikatorem struktury. Wartość jest następująca:

MQPSXP_STRUCID

MQPSXP_STRUCID to identyfikator struktury parametru wyjścia publikowania. W przypadku języka programowania w języku C stała MQPSXP_STRUC_ID_ARRAY jest również zdefiniowana; ma taką samą wartość jak MQPSXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID jest polem wejściowym do wyjścia.

Version (MQLONG)

Version jest numerem wersji struktury. Wartość jest następująca:

MQPSXP_VERSION_1

MQPSXP_VERSION_1 jest strukturą parametru wyjścia publikowania w wersji 1. Stała MQPSXP_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version jest polem wejściowym do wyjścia.

ExitId (MQLONG)

ExitId jest typem wywołanego wyjścia. Wartość jest następująca:

MQXT_PUBLISH_EXIT

Publikuj wyjście.

ExitId jest polem wejściowym do wyjścia.

ExitReason (MQLONG)

ExitReason jest przyczyną wywołania wyjścia. Możliwe wartości:

MQXR_INIT

Wyjście dla tego połączenia jest wywoływane w celu zainicjowania. Wyjście może uzyskać i zainicjować zasoby, których potrzebuje; na przykład pamięć główna.

MQXR_TERM

Wyjście dla tego połączenia jest wywoływane, ponieważ wyjście ma zostać zatrzymane. Program obsługi wyjścia musi zwolnić wszystkie zasoby, które zostały przez niego pozyskane od momentu jego zainicjowania, na przykład pamięć główna.

MQXR_PUBLICATION

Wyjście jest wywoływane przez menedżer kolejek, zanim opublikuje publikację w kolejce komunikatów subskrybenta. Wyjście może zmienić komunikat, nie umieścić komunikatu w kolejce lub wstrzymać publikację.

ExitReason jest polem wejściowym do wyjścia.

ExitResponse (MQLONG)

Ustaw *ExitResponse* w wyjściu, aby określić, w jaki sposób przetwarzanie musi być kontynuowane. *ExitResponse* to jedna z następujących wartości:

MQXCC_OK

Ustaw MQXCC_OK, aby kontynuować przetwarzanie normalnie. Ustaw wartość MQXCC_OK w odpowiedzi na dowolne wartości *ExitReason*.

Jeśli parametr *ExitReason* ma wartość MQXR_PUBLICATION, pola *DestinationQName* i *DestinationQMgrName* struktury MQSBC identyfikują miejsce docelowe, do którego wysyłany jest komunikat.

MQXCC_FAILED

Ustaw opcję MQXCC_FAILED, aby zatrzymać operację publikowania. Kod zakończenia MQCC_FAILED i kod przyczyny 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR są ustawione po powrocie z wyjścia.

MQXCC_SUPPRESS_FUNCTION

Ustaw MQXCC_SUPPRESS_FUNCTION, aby zatrzymać normalne przetwarzanie komunikatu. Ustaw MQXCC_SUPPRESS_FUNCTION tylko wtedy, gdy *ExitReason* ma wartość MQXR_PUBLICATION.

Komunikat jest nadal przetwarzany przez menedżer kolejek zgodnie z opcją MQRO_DISCARD_MSG w polu *Report* w deskrypcji komunikatu komunikatu.

- Jeśli zostanie podana opcja MQRO_DISCARD_MSG , komunikat nie zostanie dostarczony do subskrybenta.
- Jeśli opcja MQRO_DISCARD_MSG nie zostanie podana, komunikat zostanie umieszczony w kolejce niedostarczonych komunikatów. Jeśli nie ma kolejki niedostarczonych komunikatów lub komunikat nie może zostać pomyślnie umieszczony w kolejce niedostarczonych komunikatów, publikacja nie zostanie dostarczona do subskrybenta. Dostarczanie publikacji do innych subskrybentów zależy od wartości atrybutów obiektu tematu PMSGDLV i NPMSGDLV . Wyjaśnienie tych atrybutów znajduje się w opisach parametrów komendy DEFINE TOPIC (DEFINIOWANIE TEMATU).

ExitResponse to pole wyjściowe z wyjścia.

***ExitResponse2* (MQLONG)**

Produkt *ExitResponse2* jest zarezerwowany do użycia w przyszłości.

***Feedback* (MQLONG)**

Feedback jest kodem sprzężenia zwrotnego, który ma być używany, jeśli wyjście zwraca MQXCC_SUPPRESS_FUNCTION w *ExitResponse*.

Po wejściu do wyjścia, *Feedback* zawsze ma wartość MQFB_NONE. Jeśli wyjście zwraca MQXCC_SUPPRESS_FUNCTION, ustaw *Feedback* na wartość, która ma być używana dla komunikatu, gdy menedżer kolejek umieszcza go w kolejce niedostarczonych komunikatów. W przypadku powrotu z wyjścia, jeśli *Feedback* ma pierwotną wartość MQFB_NONE, menedżer kolejek ustawia *Feedback* na MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback to pole wejściowe/wyjściowe do wyjścia.

***ExitUserArea* (MQBYTE16)**

ExitUserArea to pole, które jest dostępne dla wyjścia do użycia. Każde połączenie ma osobny *ExitUserArea*. Długość *ExitUserArea* jest podawana przez MQ_EXIT_USER_AREA_LENGTH.

Pole *ExitReason* ma wartość MQXR_INIT podczas pierwszego wywołania wyjścia. *ExitUserArea* jest inicjowany do MQXUA_NONE przy pierwszym wywołaniu wyjścia dla połączenia. Kolejne zmiany w programie *ExitUserArea* są zachowywane w wywołaniach wyjścia.

ExitUserArea to pole wejściowe/wyjściowe do wyjścia.

***ExitData* (MQCHAR32)**

ExitData to stałe dane wyjściowe zdefiniowane przez parametr *PublishExitData* w sekcji w pliku inicjowania menedżera kolejek. Dane są dopełniane spacjami do pełnej długości pola. Jeśli w pliku inicjowania nie zdefiniowano żadnych stałych danych wyjściowych, pole *ExitData* jest puste. Długość *ExitData* jest podawana przez MQ_EXIT_DATA_LENGTH.

ExitData jest polem wejściowym do wyjścia.

***QMgrName* (MQCHAR48)**

QMgrName to nazwa lokalnego menedżera kolejek. Nazwa jest dopełniona spacjami do pełnej długości pola. Długość tego pola jest podawana przez produkt MQ_Q_MGR_NAME_LENGTH.

QMgrName jest polem wejściowym do wyjścia.

***Hconn* (MQHCONN)**

Hconn jest to uchwyt reprezentujący połączenie z menedżerem kolejek. Do pracy z właściwościami komunikatu można używać tylko programu *Hconn* jako parametru do wywołania funkcji właściwości komunikatu MQSETMP, MQINQMPLub MQDLTMP .

Hconn jest polem wejściowym do wyjścia.

***MsgDescPtr* (PMQMD)**

MsgDescPtr to adres deskryptora komunikatu (MQMD) przetwarzanego komunikatu i jest to kopia deskryptora MQMD zwróconego przez wywołanie MQPUT. Wyjście może zmienić zawartość deskryptora komunikatu. Każda zmiana w zawartości deskryptora komunikatu musi być wykonana z ostrożnością. W szczególności w przypadku, gdy pole *SubType* w strukturze MQSBC ma wartość MQSUBTYPE_PROXY, pole *CorrelId* w deskrypcorze komunikatu nie może być zmieniane.

No message descriptor is passed to the exit if *ExitReason* is MQXR_INIT or MQXR_TERM ; in these cases, *MsgDescPtr* is the null pointer.

MsgDescPtr jest polem wejściowym do wyjścia.

MsgHandle (MQHMSG)

MsgHandle jest to uchwyt do właściwości komunikatu. Do pracy z właściwościami komunikatu można używać tylko *MsgHandle* z wywołaniami funkcji właściwości komunikatu MQSETMP, MQINQMMP lub MQDLTMP.

MsgHandle jest polem wejściowym do wyjścia.

MsgInPtr (PMQVOID)

MsgInPtr jest adresem wejściowych danych komunikatu. Zawartość buforu adresowanego przez program *MsgInPtr* może być modyfikowana przez wyjście; patrz *MsgOutPtr*.

MsgInPtr jest polem wejściowym do wyjścia.

MsgInLength (MQLONG)

MsgInLength to długość (w bajtach) danych komunikatu przekazana do wyjścia. Adres danych jest podawany przez produkt *MsgInPtr*.

MsgInLength jest polem wejściowym do wyjścia.

MsgOutPtr (PMQVOID)

MsgOutPtr jest adresem buforu zawierającego dane komunikatu, które są zwracane z wyjścia. W przypadku wejścia do wyjścia program *MsgOutPtr* ma wartość NULL. W przypadku powrotu z wyjścia, jeśli wartość jest nadal pusta, menedżer kolejek wysyła komunikat określony przez produkt *MsgInPtr* o długości podanej w produkcie *MsgInLength*.

Jeśli wyjście modyfikuje dane komunikatu, użyj jednej z następujących procedur:

- Jeśli długość danych nie ulegnie zmianie, dane mogą być modyfikowane w buforze adresowanym przez program *MsgInPtr*. W takim przypadku nie należy zmieniać *MsgOutPtr* ani *MsgOutLength*.
- Jeśli zmodyfikowane dane są krótsze niż dane oryginalne, dane mogą być modyfikowane w buforze adresowanym przez program *MsgInPtr*. W tym przypadku wartość *MsgOutPtr* musi być ustawiona na adres buforu komunikatów wejściowych, a parametr *MsgOutLength* na nową długość danych komunikatu.
- Jeśli zmodyfikowane dane są lub mogą być dłuższe niż pierwotne dane, wyjście musi uzyskać nowy bufor komunikatów. Skopiuj do niego zmodyfikowane dane. Ustaw wartość *MsgOutPtr* na adres nowego buforu, a następnie ustaw wartość *MsgOutLength* na długość nowych danych komunikatu. Wyjście jest odpowiedzialne za zwolnienie buforu zaadresowanego przez program *MsgOutPtr* po następnym wywołaniu wyjścia.

Uwaga: *MsgOutPtr* jest zawsze pustym wskaźnikiem na wejściu do wyjścia, a nie adresem wcześniej uzyskanego buforu komunikatów. Aby zwolnić wcześniej uzyskany bufor, wyjście musi zapisać swój adres i długość. Zapisz informacje w programie *ExitUserArea* lub w bloku kontrolnym, który ma swój adres zapisany w programie *ExitUserArea*.

MsgOutPtr to pole wejściowe/wyjściowe do wyjścia.

MsgOutLength (MQLONG)

MsgOutLength to długość (w bajtach) danych komunikatu zwróconych przez wyjście. W przypadku wejścia do wyjścia to pole jest zawsze równe zero. W przypadku powrotu z wyjścia to pole jest ignorowane, jeśli parametr *MsgOutPtr* ma wartość NULL. Informacje na temat modyfikowania danych komunikatu zawiera sekcja *MsgOutPtr*.

MsgOutLength to pole wejściowe/wyjściowe do wyjścia.

pEntryPoints (PMQIEP)

pEntryPoints jest adresem struktury MQIEP, za pośrednictwem której mogą być wykonywane wywołania MQI i DCI.

Deklaracja języka C-MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4      StrucId;          /* Structure identifier */
    MQLONG       Version;          /* Structure version number */
    MQLONG       ExitId;           /* Type of exit */
    MQLONG       ExitReason;       /* Reason for invoking exit */
    MQLONG       ExitResponse;     /* Response from exit */
    MQLONG       ExitResponse2;    /* Reserved */
    MQLONG       Feedback;         /* Feedback code */
    MQBYTE16     ExitUserArea;     /* Exit user area */
    MQCHAR32     ExitData;         /* Exit data */
    MQCHAR48     QMgrName;         /* Name of local queue manager */
    MQHCONN      Hconn;           /* Connection handle */
    MQHMSG       MsgHandle;        /* Handle to message properties */
    PMQMD        MsgDescPtr;       /* Address of message descriptor */
    PMQVOID      MsgInPtr;         /* Address of input message data */
    MQLONG       MsgInLength;      /* Length of input message data */
    PMQVOID      MsgOutPtr;        /* Address of output message data */
    MQLONG       MsgOutLength;     /* Length of output message data */
    /* Ver:1 */
    PMQIEP       pEntryPoints;     /* Address of the MQIEP structure */
    /* Ver:2 */
} MQPSXP;
```

MQPBC-Struktura danych kontekstu publikowania

Struktura MQPBC zawiera informacje kontekstowe odnoszące się do publikatora publikacji, które są przekazywane do wyjścia publikowania.

Tabela 190 na stronie 1558 podsumowuje pola w strukturze:

Tabela 190. Pola w tabeli MQPBC	
Pole	Opis
<u>StrucID</u>	Identyfikator struktury
<u>Version</u>	Numer wersji struktury
<u>PubTopicString</u>	Łańcuch tematu publikacji
<u>MsgDescPtr</u>	Adres deskryptora komunikatu (MQMD)

Pola

StrucID (MQCHAR4)

StrucID jest identyfikatorem struktury. Wartość jest następująca:

MQPBC_STRUCID

MQPBC_STRUCID to identyfikator struktury kontekstu publikacji. W przypadku języka programowania w języku C stała MQPBC_STRUC_ID_ARRAY jest również zdefiniowana; ma taką samą wartość jak MQPBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID jest polem wejściowym do wyjścia.

Version (MQLONG)

Version jest numerem wersji struktury. Wartość jest następująca:

MQPBC_VERSION_1

MQPBC_VERSION_1 jest strukturą parametru wyjścia publikowania w wersji 1.

MQPBC_VERSION_2

MQPBC_VERSION_2 jest strukturą parametru wyjścia publikowania w wersji 2. Stała MQPBC_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version jest polem wejściowym do wyjścia.

PubTopicString (MQCHARV)

PubTopicString to łańcuch tematu, w którym jest publikowany łańcuch tematu.

PubTopicString jest polem wejściowym do wyjścia.

MsgDescPtr (PMQMD)

MsgDescPtr jest adresem kopii deskryptora komunikatu (MQMD) dla przetwarzanego komunikatu.

MsgDescPtr jest polem wejściowym do wyjścia.

Deklaracja języka C-MQPBC

```
typedef struct tagMQPBC {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQCHARV    PubTopicString;   /* Publish topic string */  
    PMQMD      MsgDescPtr;       /* Address of message descriptor */  
} MQPBC;
```

MQSBC-Struktura danych kontekstu subskrypcji

Struktura MQSBC zawiera informacje kontekstowe odnoszące się do subskrybenta, który odbiera publikację, która jest przekazywana do wyjścia publikowania.

Tabela 191 na stronie 1559 podsumowuje pola w strukturze:

Tabela 191. Zmienne w MQSBC	
Pole	Opis
<u>StrucID</u>	Identyfikator struktury
<u>Version</u>	Numer wersji struktury
<u>DestinationQMgrName</u>	Nazwa docelowego menedżera kolejek
<u>DestinationQName</u>	Nazwa kolejki docelowej
<u>SubType</u>	Typ subskrypcji.
<u>SubOptions</u>	Opcje subskrypcji
<u>ObjectName</u>	Nazwa obiektu
<u>ObjectString</u>	Łańcuch obiektu
<u>SubTopicString</u>	Łańcuch tematu subskrypcji
<u>SubName</u>	Nazwa subskrypcji
<u>SubId</u>	Identyfikator subskrypcji
<u>SelectionString</u>	Adres łańcucha wyboru
<u>SubLevel</u>	Poziom subskrypcji
<u>PSProperties</u>	Właściwości publikowania/subskrybowania

Pola

StrucID (MQCHAR4)

Identyfikator struktury. Wartość jest następująca:

MQSBC_STRUCID

MQSBC_STRUCID to identyfikator struktury parametru wyjścia publikowania. Dla języka programowania w języku C stała MQSBC_STRUC_ID_ARRAY jest również zdefiniowana;

MQSBC_STRUC_ID_ARRAY ma taką samą wartość jak MQSBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID jest polem wejściowym do wyjścia.

Version (MQLONG)

Numer wersji struktury. Wartość jest następująca:

MQSBC_VERSION_1

Struktura parametru wyjścia publikowania wersji 1. Stała MQSBC_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version jest polem wejściowym do wyjścia.

DestinationQMGrName (MQCHAR48)

DestinationQMGrName to nazwa menedżera kolejek, do którego wysyłany jest komunikat. Nazwa jest dopełniona spacjami do pełnej długości pola. Nazwa może zostać zmieniona przez wyjście. Długość tego pola jest podawana przez produkt MQ_Q_MGR_NAME_LENGTH.

DestinationQMGrName to pole wejściowe/wyjściowe do wyjścia; patrz [uwaga](#).

DestinationQName (MQCHAR48)

DestinationQName to nazwa kolejki, do której wysyłany jest komunikat. Nazwa jest dopełniona spacjami do pełnej długości pola. Nazwa może zostać zmieniona przez wyjście. Długość tego pola jest podawana przez produkt MQ_Q_NAME_LENGTH.

DestinationQName to pole wejściowe/wyjściowe do wyjścia; patrz [uwaga](#).

SubType (MQLONG)

SubType wskazuje, w jaki sposób subskrypcja została utworzona. Poprawne wartości to MQSUBTYPE_API, MQSUBTYPE_ADMIN i MQSUBTYPE_PROXY . patrz [Inquire Subscription Status \(Response\)](#)(Status subskrypcji zapytania).

SubType jest polem wejściowym do wyjścia.

SubOptions (MQLONG)

SubOptions to opcje subskrypcji; patrz [“Opcje \(MQLONG\)” na stronie 571](#) , aby uzyskać opis wartości, które może przyjąć to pole.

SubOptions jest polem wejściowym do wyjścia.

ObjectName (MQCHAR48)

ObjectName to nazwa obiektu tematu zgodnie z definicją w lokalnym menedżerze kolejek. Długość tego pola jest podawana przez produkt MQ_TOPIC_NAME_LENGTH. Nazwa obiektu to nazwa obiektu tematu administracyjnego, który menedżer kolejek powiązany jest z łańcuchem tematu. Nawet jeśli subskrybent udostępnił obiekt tematu jako część subskrypcji, *ObjectName* może być innym obiektem tematu. Powiązanie obiektu tematu z subskrypcją jest zależne od pełnej rozdzielczości produktu *SubTopicString*.

ObjectName jest polem wejściowym do wyjścia.

ObjectString (MQCHARV)

ObjectString to pełny łańcuch tematu publikacji, która została zasubskrybowana. Wszystkie znaki wieloznaczne w oryginalnym łańcuchu subskrypcji są rozstrzygane. It is different to the MQSD subscription *ObjectString* field described in [“ObjectString \(MQCHARV\)” na stronie 571](#), which might contain wildcards, and is exclusive of any object name provided by the subscriber.

ObjectString jest polem wejściowym do wyjścia.

SubTopicString (MQCHARV)

SubTopicString jest kompletnym łańcuchem tematu dostarczonemu przez subskrybenta. *SubTopicString* jest kombinacją łańcucha tematu zdefiniowanego w obiekcie tematu oraz łańcucha tematu. Subskrybent musi udostępniać obiekt tematu, łańcuch tematu lub oba te elementy. Jeśli subskrybent udostępnia łańcuch tematu, może on zawierać znaki wieloznaczne.

SubTopicString jest polem wejściowym do wyjścia.

SubName (MQCHARV)

SubName to nazwa subskrypcji, która jest udostępniana przez subskrybenta lub jest nazwą wygenerowaną.

SubName jest polem wejściowym do wyjścia.

SubId (MQBYTE 24)

SubId jest unikalnym wewnętrznym identyfikatorem subskrypcji.

SubId jest polem wejściowym do wyjścia.

SelectionString (MQCHARV)

SelectionString to kryteria wyboru używane podczas subskrybowania komunikatów z tematu. Patrz sekcja [Selektory](#).

SelectionString jest polem wejściowym do wyjścia.

SubLevel (MQLONG)

SubLevel to poziom przechwytywania powiązany z subskrypcją. Więcej informacji na ten temat zawiera sekcja [“SubLevel \(MQLONG\)”](#) na stronie 583 .

SubLevel jest polem wejściowym do wyjścia.

PSPProperties (MQLONG)

PSPProperties to właściwości publikowania/subskrypcji. Określają, w jaki sposób właściwości komunikatu związane z publikowaniem/subskrybowaniem są dodawane do komunikatów wysyłanych do tej subskrypcji. Możliwe wartości to: MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Opis tych wartości znajduje się w sekcji [Parametry opcjonalne \(zmiana, kopiowanie i tworzenie subskrypcji\)](#) .

PSPProperties jest polem wejściowym do wyjścia.

Uwaga: Sprawdzenia autoryzacji są wykonywane tylko na oryginalnych wartościach

DestinationQMgrName i *DestinationQName* , zanim zostaną przekazane do wyjścia publikowania. Żadne nowe sprawdzenia autoryzacji nie są wykonywane, gdy wyjście zmienia kolejkę docelową. W tym celu należy zmienić *DestinationQMgrName* lub *DestinationQName*.

Deklaracja języka C-MQSBC

```
typedef struct tagMQSBC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQCHAR48    DestinationQMgrName; /* Destination queue manager */
    MQCHAR48    DestinationQName;  /* Destination queue name */
    MQLONG     SubType;           /* Type of subscription */
    MQLONG     SubOptions;         /* Subscription options */
    MQCHAR48    ObjectName;        /* Object name */
    MQCHARV     ObjectString;       /* Object string */
    MQCHARV     SubTopicString;     /* Subscription topic string */
    MQCHARV     SubName;           /* Subscription name */
    MQBYTE24    SubId;            /* Subscription identifier */
    MQCHARV     SelectionString;    /* Subscription selection string */
    MQLONG     SubLevel;          /* Subscription level */
    MQLONG     PSPProperties;       /* Publish/subscribe properties */
} MQSBC;
```

Wywołania wyjścia kanału i struktury danych

Ta kolekcja tematów zawiera informacje uzupełniające na temat specjalnych wywołań IBM MQ i struktur danych, które mogą być używane podczas pisania programów obsługi wyjścia kanału.

Informacje te są informacjami o interfejsie programistycznym wrażliwym na produkt. Istnieje możliwość pisania wyjść użytkownika programu IBM MQ w następujących językach programowania:

Platforma	Języki programowania
IBM MQ for z/OS	Asembler i C (które muszą być zgodne ze środowiskiem programistycznym systemu C dla wyjść systemowych, opisane w podręczniku <i>z/OS C/C++ Programming Guide</i>).
IBM MQ for IBM i	ILE C, ILE COBOL i ILE RPG
Wszystkie inne platformy IBM MQ	C

Istnieje również możliwość pisania wyjść użytkownika w produkcie Java do użycia tylko z aplikacjami Java i JMS. Więcej informacji na temat tworzenia i używania wyjść kanału z IBM MQ classes for Java można znaleźć w sekcji [Używanie wyjść kanału w produkcie IBM MQ classes for Java](#) oraz w systemie IBM MQ classes for JMS, patrz sekcja [Używanie wyjść kanału z programem IBM MQ classes for JMS](#).

Nie można zapisywać wyjść użytkownika IBM MQ w języku TAL ani Visual Basic. Jednak deklaracja dla struktury MQCD jest udostępniona w języku Visual Basic do użycia w wywołaniu MQCONN z programu IBM MQ MQI client.

W wielu przypadkach w opisach, które są zgodne, parametry są tablicami lub łańcuchami znaków o rozmiarze, który nie jest ustalony. W przypadku tych parametrów do reprezentowania stałej liczbowej używana jest mała litera "n". Jeśli deklaracja dla tego parametru jest zakodowana, wartość "n" musi być zastąpiona wartością liczbową wymaganą. Więcej informacji na temat konwencji używanych w tych opisach można znaleźć w publikacji ["Elementarne typy danych"](#) na stronie 237.

Pliki definicji danych

Pliki definicji danych są dostarczane wraz z produktem IBM MQ dla każdego z obsługiwanych języków programowania. Szczegółowe informacje na temat tych plików można znaleźć w sekcji [Kopiowanie, nagłówki, dołączanie i pliki modułów](#).

MQ_CHANNEL_EXIT-wyjście kanału

Wywołanie MQ_CHANNEL_EXIT opisuje parametry, które są przekazywane do każdego wyjścia kanału wywołanego przez agenta kanału komunikatów.

Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_CHANNEL_EXIT. Nazwa MQ_CHANNEL_EXIT nie ma specjalnego znaczenia, ponieważ nazwy wyjść kanału są udostępniane w definicji kanału MQCD.

Istnieje pięć typów wyjścia kanału:

- Wyjście zabezpieczeń kanału
- Wyjście komunikatu kanału
- Wyjście wysyłania kanału
- Wyjście odbierania kanału
- Komunikat kanału-wyjście ponowienia

Parametry są podobne dla każdego typu wyjścia, a opis podany w tym miejscu dotyczy wszystkich, z wyjątkiem sytuacji, w których zaznaczono inaczej.

Składnia

MQ_CHANNEL_EXIT (*ChannelExitParms*, *ChannelDefinition*, *DataLength*, *AgentBufferLength*, *AgentBuffer*, *ExitBufferLength*, *ExitBufferAddr*)

Parametry

Wywołanie MQ_CHANNEL_EXIT ma następujące parametry.

ChannelExitParms (MQCXP)-wejście/wyjście

Blok parametru wyjścia kanału.

Struktura ta zawiera dodatkowe informacje związane z wywoływaniem wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać, jak działa agent MCA.

ChannelDefinition (MQCD)-wejście/wyjście

Definicja kanału.

Struktura ta zawiera parametry ustawione przez administratora w celu sterowania zachowaniem kanału.

DataLength (MQLONG)-input/output

Długość danych.

Dane zależą od typu wyjścia:

- W przypadku wyjścia zabezpieczeń kanału, po wywołaniu procedury zewnętrznej ten parametr zawiera długość dowolnego komunikatu zabezpieczeń w polu *AgentBuffer*, jeśli parametr *ExitReason* ma wartość MQXR_SEC_MSG. Wartość zero, jeśli nie ma komunikatu. Wyjście musi ustawić to pole na długość dowolnego komunikatu zabezpieczeń, który ma zostać wysłany do jego partnera, jeśli jest ustawiony na wartość *ExitResponse* na wartość MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG. Dane komunikatu znajdują się w *AgentBuffer* lub *ExitBufferAddr*.

Treść wiadomości o bezpieczeństwie jest wyłączną odpowiedzialnością za wyjścia bezpieczeństwa.

- W przypadku wyjścia komunikatu kanału, po wywołaniu wyjścia ten parametr zawiera długość komunikatu (wraz z nagłówkiem kolejki transmisji). Wyjście musi ustawić to pole na długość komunikatu w produkcie *AgentBuffer* lub *ExitBufferAddr*, który ma być kontynuowany. Wartość ta musi być większa lub równa długości nagłówka kolejki transmisji (MQXQH).
- W przypadku wyjścia odbierania kanału lub kanału odbierania kanału, po wywołaniu wyjścia ten parametr zawiera długość transmisji. Wyjście musi ustawić to pole na długość transmisji w produkcie *AgentBuffer* lub *ExitBufferAddr*, który ma być kontynuowany.

Jeśli wyjście zabezpieczeń wysyła komunikat, a na drugim końcu kanału nie ma wyjścia zabezpieczeń, lub drugi koniec ustawia *ExitResponse* dla MQXCC_OK, wyjście inicjujące jest ponownie wywoływane z MQXR_SEC_MSG i pustą odpowiedzią (*DataLength* = 0).

AgentBufferLength (MQLONG)-dane wejściowe

Długość buforu agenta.

Ten parametr może być większy niż parametr *DataLength* w wywołaniu.

W przypadku komunikatów kanału, wysyłania i odbierania wszystkie nieużywane miejsca w wywołaniu może być używane przez wyjście w celu rozszerzenia danych w lokalizacji. Jeśli tak się stanie, parametr *DataLength* musi być odpowiednio ustawiony przez wyjście.

W języku programowania C ten parametr jest przekazywany przez adres.

AgentBuffer (MQBYTE x AgentBufferLength)-input/output

Bufer agenta.

Zawartość tego parametru zależy od typu wyjścia:

- W przypadku wyjścia zabezpieczeń kanału, w przypadku wywołania wyjścia, zawiera on komunikat zabezpieczeń, jeśli *ExitReason* to MQXR_SEC_MSG. Aby wysłać komunikat bezpieczeństwa z powrotem, wyjście może albo użyć tego buforu, albo własnego buforu (*ExitBufferAddr*).
- W przypadku wyjścia komunikatu kanału, w wywołaniu wyjścia z tego parametru znajdują się:
 - Nagłówek kolejki transmisji (MQXQH), który zawiera deskryptor komunikatu (który sam zawiera informacje o kontekście dla komunikatu), po czym następuje bezpośrednio po nim
 - Dane komunikatu

Jeśli komunikat ma być kontynuowany, wyjście może wykonać jedną z następujących czynności:

- Pozostaw zawartość buforu bez zmian
- Zmodyfikuj zawartość w lokalizacji (zwracając nową długość danych w produkcie *DataLength*). to nie może być większe niż *AgentBufferLength*)
- Skopiuj zawartość do *ExitBufferAddr*, dokonaj wszelkich wymaganych zmian

Nie są sprawdzane wszystkie zmiany wprowadzone przez wyjście do nagłówka kolejki transmisji. Jednak błędne modyfikacje mogą oznaczać, że komunikat nie może zostać umieszczony w miejscu docelowym.

- W przypadku wyjścia wysyłania lub odbierania kanału, po wywołaniu wyjścia zawiera on dane transmisji. Wyjście może wykonać jedną z następujących czynności:
 - Pozostaw zawartość buforu bez zmian
 - Zmodyfikuj zawartość w lokalizacji (zwracając nową długość danych w produkcie *DataLength*). to nie może być większe niż *AgentBufferLength*)
 - Skopiuj zawartość do *ExitBufferAddr*, dokonaj wszelkich wymaganych zmian

Pierwsze 8 bajtów danych nie może być zmieniane przez wyjście.

ExitBufferLength (MQLONG)-input/output

Długość buforu wyjściowego.

W przypadku pierwszego wywołania wyjścia ten parametr jest ustawiony na zero. Po tym czasie każda wartość jest przekazywana z powrotem przez wyjście, przy każdym wywołaniu, jest przedstawiana do wyjścia przy następnym wywołaniu. Wartość nie jest używana przez agenta MCA.

Uwaga: Parametr ten nie może być używany przez wyjścia zapisane w językach programowania, które nie obsługują typu danych wskaźnika.

ExitBufferAddr (MQPTR)-wejście/wyjście

Adres buforu wyjścia.

Ten parametr jest wskaźnikiem do adresu buforu pamięci masowej zarządzanego przez wyjście, w którym może on zwracać dane komunikatu lub transmisji (w zależności od typu wyjścia) do agenta, jeśli bufor agenta jest lub może nie być wystarczająco duży, lub jeśli jest wygodniejszy dla wyjścia, aby to zrobić.

Przy pierwszym wywołaniu wyjścia adres przekazany do wyjścia ma wartość NULL. Po tym, jaki adres zostanie przekazany z powrotem przez wyjście, w każdym wywołaniu zostanie wyświetlone wyjście przy następnym wywołaniu.

Jeśli parametr *ExitBufferAddr* ma wartość NULL, używane dane są pobierane z parametru *AgentBuffer*.

Jeśli parametr *ExitBufferAddr* nie ma wartości NULL, używane dane są pobierane z buforu wskazanego przez parametr *Addr ExitBuffer*.

Uwaga: Ten parametr nie może być używany przez wyjścia zapisane w językach programowania, które nie obsługują typu danych wskaźnika.

Wywołanie C

```
exitname (&ChannelExitParms, &ChannelDefinition,  
&DataLength, &AgentBufferLength, AgentBuffer,  
&ExitBufferLength, &ExitBufferAddr);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
MQCXP  ChannelExitParms;  /* Channel exit parameter block */  
MQCD   ChannelDefinition; /* Channel definition */
```



```

MQLONG  DataLength;          /* Length of data */
MQLONG  AgentBufferLength;   /* Length of agent buffer */
MQBYTE  AgentBuffer[n];      /* Agent buffer */
MQLONG  ExitBufferLength;    /* Length of exit buffer */
MQPTR   ExitBufferAddr;      /* Address of exit buffer */

```

Wywołanie języka COBOL

```

CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                      DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                      EXITBUFFERLENGTH, EXITBUFFERADDR.

```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```

** Channel exit parameter block
01 CHANNELEXITPARMS.
   COPY CMQCXPV.
** Channel definition
01 CHANNELDEFINITION.
   COPY CMQCDV.
** Length of data
01 DATALENGTH      PIC S9(9) BINARY.
** Length of agent buffer
01 AGENTBUFFERLENGTH PIC S9(9) BINARY.
** Agent buffer
01 AGENTBUFFER       PIC X(n).
** Length of exit buffer
01 EXITBUFFERLENGTH  PIC S9(9) BINARY.
** Address of exit buffer
01 EXITBUFFERADDR    POINTER.

```

Wywołanie RPG (ILE)

```

C*..1.....2.....3.....4.....5.....6.....7..
C          CALLP      exitname(MQCXP : MQCD : DATLEN :
C                               ABUFL : ABUF : EBUFL :
C                               EBUF)

```

Definicja prototypu dla wywołania to:

```

D*..1.....2.....3.....4.....5.....6.....7..
Dexitname      PR          EXTPROC('exitname')
D* Channel exit parameter block
D MQCXP                160A
D* Channel definition
D MQCD                 1328A
D* Length of data
D DATLEN                10I 0
D* Length of agent buffer
D ABUFL                 10I 0
D* Agent buffer
D ABUF                  *   VALUE
D* Length of exit buffer
D EBUFL                 10I 0
D* Address of exit buffer
D EBUF                  *

```

Wywołanie asemblera System/390

```

CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION, DATALENGTH, X
               AGENTBUFFERLENGTH, AGENTBUFFER, EXITBUFFERLENGTH, X
               EXITBUFFERADDR)

```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

CHANNELEXITPARMS	CMQXPA	,	Channel exit parameter block
CHANNELDEFINITION	CMQCDA	,	Channel definition
DATALength	DS	F	Length of data
AGENTBUFFERLENGTH	DS	F	Length of agent buffer
AGENTBUFFER	DS	CL(n)	Agent buffer
EXITBUFFERLENGTH	DS	F	Length of exit buffer
EXITBUFFERADDR	DS	F	Address of exit buffer

Użycie notatek

1. Funkcja wykonywana przez wyjście kanału jest zdefiniowana przez dostawcę wyjścia. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w tym miejscu i w powiązonym bloku kontrolnym, MQCXP.
2. Parametr *ChannelDefinition* przekazany do wyjścia kanału może być jedną z kilku wersji. Więcej informacji na ten temat zawiera pole *Version* w strukturze MQCD.
3. Jeśli wyjście kanału odbierze strukturę MQCD z polem *Version* ustawionym na wartość większą niż MQCD_VERSION_1, wyjście musi używać pola *ConnectionName* z tabeli MQCD, w preferencjach do pola *ShortConnectionName*.
4. W ogólnym przypadku wyjścia kanału są dozwolone w celu zmiany długości danych komunikatu. Może się to pojawić w wyniku wyjścia z dodania danych do komunikatu lub usunięcia danych z komunikatu lub do kompresowania lub szyfrowania komunikatu. Jednak, jeśli komunikat jest segmentem zawierającym tylko część komunikatu logicznego, mają zastosowanie ograniczenia specjalne. W szczególności, w związku z działaniami uzupełniającymi się i odbierającymi wyjściami, nie może być żadnych zmian netto w długości komunikatu.

Na przykład, dozwolone jest wysłanie wyjścia wysyłającego w celu skrócenia komunikatu przez jego kompresowanie, ale komplementarne wyjście odbierające musi przywrócić pierwotną długość komunikatu przez dekompresowanie go, tak aby nie było żadnej zmiany sieci w długości komunikatu.

To ograniczenie wynika z faktu, że zmiana długości segmentu spowoduje, że przesunięcia późniejszych segmentów w komunikacie są niepoprawne, a to uniemożliwiłoby menedżerowi kolejek rozpoznanie, że segmenty utworzyły kompletny komunikat logiczny.

MQ_CHANNEL_AUTO_DEF_EXIT-wyjście automatyczne definicji kanału

Wywołanie MQ_CHANNEL_AUTO_DEF_EXIT opisuje parametry, które są przekazywane do wyjścia automatycznego definiowania kanału wywołanego przez agenta kanału komunikatów.

Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_CHANNEL_AUTO_DEF_EXIT. Nazwa MQ_CHANNEL_AUTO_DEF_EXIT nie ma specjalnego znaczenia, ponieważ w menedżerze kolejek są udostępnione nazwy wyjść automatycznej definicji.

Składnia

MQ_CHANNEL_AUTO_DEF_EXIT (*ChannelExitParms*, *ChannelDefinition*)

Parametry

Wywołanie MQ_CHANNEL_AUTO_DEF_EXIT ma następujące parametry.

ChannelExitParms (MQCXP)-wejście/wyjście

Blok parametru wyjścia kanału.

Struktura ta zawiera dodatkowe informacje związane z wywoływaniem wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać, jak działa agent MCA.

ChannelDefinition (MQCD)-wejście/wyjście

Definicja kanału.

Struktura ta zawiera parametry ustawione przez administratora w celu sterowania zachowaniem kanałów, które są tworzone automatycznie. Wyjście ustawia informacje w tej strukturze w celu zmodyfikowania domyślnego zachowania ustawionego przez administratora.

Wymienione pola MQCD nie mogą być zmieniane przez wyjście:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Jeśli inne pola zostaną zmienione, wartość ustawiona przez wyjście musi być poprawna. Jeśli wartość nie jest poprawna, komunikat o błędzie jest zapisywany w pliku dziennika błędów lub jest wyświetlany na konsoli (w zależności od środowiska).



Ostrzeżenie: Automatycznie zdefiniowane kanały utworzone przez wyjście CHAD (channel automatic definition) nie mogą ustawić etykiety certyfikatu, ponieważ uzgadnianie SSL lub TLS wystąpiło podczas tworzenia kanału. Ustawienie etykiety certyfikatu w wyjściu CHAD dla kanałów przychodzących nie ma żadnego efektu.

Wywołanie C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
MQCXP  ChannelExitParms;  /* Channel exit parameter block */
MQCD   ChannelDefinition; /* Channel definition */
```

Wywołanie języka COBOL

```
CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
** Channel exit parameter block
01 CHANNELEXITPARMS.
   COPY CMQCXPV.
** Channel definition
01 CHANNELDEFINITION.
   COPY CMQCDV.
```

Wywołanie RPG (ILE)

```
C*.1.....2.....3.....4.....5.....6.....7..
C                                CALLP      exitname(MQCXP : MQCD)
```

Definicja prototypu dla wywołania to:

```
D*.1.....2.....3.....4.....5.....6.....7..
Dexitname          PR              EXTPROC('exitname')
D* Channel exit parameter block
D MQCXP              160A
D* Channel definition
D MQCD              1328A
```

Wywołanie asemblera System/390

```
CALL EXITNAME, (CHANNELEXITPARMS, CHANNELDEFINITION)
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
CHANNELEXITPARMS  CMQXCPA  , Channel exit parameter block
CHANNELDEFINITION CMQCDA   , Channel definition
```

Użycie notatek

1. Funkcja wykonywana przez wyjście kanału jest zdefiniowana przez dostawcę wyjścia. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w tym miejscu i w powiązonym bloku kontrolnym, MQCXP.
2. Parametr *ChannelExitParms* przekazany do wyjścia automatycznego definiowania kanału jest strukturą MQCXP. Przekazana wersja programu MQCXP zależy od środowiska, w którym działa wyjście. Szczegółowe informacje można znaleźć w opisie pola *Version* w [“MQCXP-parametr wyjścia kanału” na stronie 1610](#).
3. Parametr *ChannelDefinition* przekazany do wyjścia automatycznego definiowania kanału jest strukturą MQCD. Wersja przekazanego produktu MQCD zależy od środowiska, w którym jest uruchomiony program obsługi wyjścia. Szczegółowe informacje zawiera opis pola *Version* w publikacji [“MQCD-definicja kanału” na stronie 1569](#).

MQXWAIT-oczekiwanie na zakończenie

Wywołanie MQXWAIT oczekuje na wystąpienie zdarzenia. Może być używany tylko z poziomu wyjścia kanału w systemie z/OS.

Użycie komendy MQXWAIT pomaga uniknąć problemów z wydajnością, które w przeciwnym razie mogą wystąpić, jeśli wyjście kanału wykonuje coś, co powoduje oczekiwanie. Zdarzenie MQXWAIT oczekuje na sygnalizację przez system MVS ECB (blok kontrolny zdarzeń). ECB jest opisany w opisie bloku kontrolnego MQXWD.

 Więcej informacji na temat korzystania z programu MQXWAIT i pisania programów obsługi wyjścia kanału zawiera sekcja [Zapisywanie programów obsługi wyjścia kanału w systemie z/OS](#).

Składnia

MQXWAIT (*Hconn*, *WaitDesc*, *CompCode*, *Reason*)

Parametry

Wywołanie MQXWAIT ma następujące parametry.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN wydane w tym samym lub wcześniejszym wywołaniu wyjścia.

WaitDesc (MQXWD)-input/output

Deskryptor oczekiwania.

Ten parametr opisuje zdarzenie, które ma czekać. Szczegółowe informacje na temat pól w tej strukturze można znaleźć w sekcji [“MQXWD-deskryptor oczekiwania wyjścia” na stronie 1625](#).

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jeden z następujących kodów:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000') Nie ma powodu do zgłoszenia.

MQRC_ADAPTER_NOT_AVAILABLE

(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_OPTIONS_ERROR

(2046, X'7FE') Opcje nie są poprawne lub niespójne.

MQRC_XWAIT_ANULOWANA

(2107, X'83B') Wywołanie MQXWAIT zostało anulowane.

BŁĄD MQRC_XWAIT_ERROR

(2108, X'83C') Wywołanie wywołania MQXWAIT nie jest poprawne.

Wywołanie C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;      /* Connection handle */
MQXWD     WaitDesc;   /* Wait descriptor */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

Wywołanie asemblera System/390

```
CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

HCONN	DS	F	Connection handle
WAITDESC	CMQXWDA	,	Wait descriptor
COMPCODE	DS	F	Completion code
REASON	DS	F	Reason code qualifying COMPCODE

MQCD-definicja kanału

Struktura MQCD zawiera parametry sterujące wykonywaniem kanału. Jest on przekazywany do każdego wyjścia kanału wywołanego przez agenta kanału komunikatów (Message Channel Agent-MCA).

Więcej informacji na temat wyjść kanału zawiera sekcja [“MQ_CHANNEL_EXIT-wyjście kanału”](#) na stronie 1562. Opis w tym temacie odnosi się zarówno do kanałów komunikatów, jak i do kanałów MQI.

Pola nazwy wyjścia

Po wywołaniu wyjścia odpowiednie pole z *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit* i *MsgRetryExit* zawiera nazwę aktualnie wywoływanej procedury zewnętrznej. Znaczenie

nazwy w tych polach zależy od środowiska, w którym działa agent MCA. Nazwa jest wyrównana do lewej w obrębie pola, bez odstępów osadzonych. Nazwa jest dopełniona spacjami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

Systemy UNIX

Nazwa wyjścia to nazwa dynamicznie ładowanego modułu lub biblioteki, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[ path ] library ( function )
```

Długość nazwy jest ograniczona do 128 znaków.

z/OS

Nazwa wyjścia to nazwa modułu ładowalnego, który jest poprawny dla specyfikacji w parametrze EP makra LINK lub LOAD. Nazwa jest ograniczona do maksymalnie ośmiu znaków.

Windows

Nazwa wyjścia jest nazwą biblioteki dołączanej dynamicznie, z przyrostkiem nazwy funkcji rezydujących w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu i napędem:

```
[d:][ path ] library ( function )
```

Długość nazwy jest ograniczona do 128 znaków.

IBM i

Nazwa wyjścia to 10-bajtowa nazwa programu, po której następuje 10-bajtowa nazwa biblioteki. Jeśli długość nazw jest mniejsza niż 10 bajtów, każda nazwa jest dopełniona spacjami, co powoduje, że jest ona 10 bajtów. Nazwą biblioteki może być *LIBL, z wyjątkiem wywołania wyjścia automatycznego definiowania kanału, w którym to przypadku wymagana jest pełna nazwa.

Zmiana pól MQCD w wyjściu kanału

Wyjście kanału może zmienić pola na zmaterializowanych tabelach MQCD. Zmieniona wartość pozostaje na zmaterializowanych tabelach MQCD i jest przekazywana do pozostałych wyjść w łańcuchu wyjścia i do dowolnej konwersacji współużytkującej instancję kanału. Zmieniona tabela MQCD jest również używana przez agenta MCA w celu normalnego przetwarzania w trakcie trwającego czasu życia kanału.

Następujące pola MQCD nie mogą być zmieniane przez wyjście:

- ChannelName
- ChannelType
- StrucLength
- Wersja

Odsyłacze pokrewne

[“Pola” na stronie 1571](#)

Ten temat zawiera listę wszystkich pól w strukturze MQCD i opisuje poszczególne pola.

[“Deklaracja C” na stronie 1597](#)

Ta deklaracja jest deklaracją języka C dla struktury MQCD.

[“Deklaracja języka COBOL” na stronie 1599](#)

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCD.

[“Deklaracja RPG \(ILE\)” na stronie 1601](#)

Ta deklaracja jest deklaracją RPG dla struktury MQCD.

[“Deklaracja assemblera System/390” na stronie 1604](#)

Ta deklaracja jest deklaracją assemblera System/390 dla struktury MQCD.

“Wizualna deklaracja podstawowa” na stronie 1606

Ta deklaracja jest deklaracją Visual Basic struktury MQCD.

“Zmiana pól MQCD w wyjściu kanału” na stronie 1607

Wyjście kanału może zmienić pola na zmaterializowanych tabelach MQCD. Zmiany te nie są jednak zazwyczaj wykonywane, z wyjątkiem sytuacji wymienionych.

Pola

Ten temat zawiera listę wszystkich pól w strukturze MQCD i opisuje poszczególne pola.

Limit BatchData(MQLONG)

To pole określa limit (w kilobajtach) ilości danych, które mogą zostać wysłane za pośrednictwem kanału przed przejnieniem punktu synchronizacji.

Punkt synchronizacji jest pobierany po przejściu przez kanał komunikatu, który spowodował osiągnięcie limitu.

Zadanie wsadowe jest przerywane, gdy spełniony zostaje jeden z następujących warunków:

- Komunikaty produktu **BatchSize** zostały wysłane.
- Liczba wysłanych bajtów: **BatchDataLimit**.
- Kolejka transmisji jest pusta, a wartość **BatchInterval** została przekroczone.

Wartość musi być z zakresu od 0 do 999999. Wartość domyślna to 5000.

Wartość zero w tym atrybucie oznacza, że żaden limit danych nie jest stosowany do zadań wsadowych w tym kanale.

Ten parametr ma zastosowanie tylko do kanałów z parametrem *ChannelType* o wartości MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSRCVR lub MQCHT_CLUSSDR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_11.

BatchHeartbeat (MQLONG)

To pole określa przedział czasu, który jest używany do wyzwolenia pulsu przetwarzania wsadowego dla kanału.

Pulsowanie wsadowe umożliwia kanałom nadawczym określenie, czy instancja kanału zdalnego jest nadal aktywna przed wątpliwością. Puls wsadowy występuje wtedy, gdy kanał nadawczy nie został skomunikowany z instancją kanału zdalnego w określonym przedziale czasu.

Wartość mieści się w zakresie od 0 do 999 999; jednostką są milisekundy. Wartość zero oznacza, że pulsowanie wsadowe nie jest włączone.

To pole jest istotne tylko dla kanałów, które mają *ChannelType* z tabeli MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

BatchInterval (MQLONG)

To pole określa przybliżony czas (w milisekundach), przez który kanał zachowuje otwartą partię, jeśli w bieżącym zadaniu wsadowym przekazano mniej komunikatów niż *BatchSize*.

Jeśli wartość *BatchInterval* jest większa od zera, to zadanie wsadowe zostaje zakończone przez którykolwiek z następujących zdarzeń, które wystąpią jako pierwsze:

- Wysłane zostały komunikaty produktu *BatchSize*, lub
- Od początku zadania wsadowego upłynęło *BatchInterval* milisekund.

Jeśli parametr *BatchInterval* ma wartość zero, zadanie wsadowe zostaje zakończone w zależności od tego, który z następujących zdarzeń wystąpi jako pierwszy:

- Wysłane zostały komunikaty produktu *BatchSize*, lub

- kolejka transmisji staje się pusta.

Wartość *BatchInterval* musi być z zakresu od zera do 999 999 999.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, gdy wartość *Version* jest mniejsza niż MQCD_VERSION_4.

BatchSize (MQLONG)

To pole określa maksymalną liczbę komunikatów, które mogą zostać wysłane za pośrednictwem kanału przed synchronizacją kanału.

To pole nie ma znaczenia dla kanałów z *ChannelType* z MQCHT_SVRCONN lub MQCHT_CLNTCONN.

CertificateLabel (MQCHAR64)

W tym polu podano szczegółowe informacje na temat używanej etykiety certyfikatu.

Program IBM MQ inicjuje wartość domyślną dla pola *CertificateLabel* jako odstęp.

Wartość ta jest interpretowana w czasie wykonywania jako wartość domyślna i jest zgodna wstecznie.

Na przykład określenie wersji MQCD mniejszej niż 11 lub użycie wartości domyślnej odstępów dla pola *CertificateLabel* oznacza, że to pole jest ignorowane.

Długość tego pola jest podana przez MQ_CERT_LABEL_LENGTH.

ChannelMonitoring (MQLONG)

To pole określa bieżący poziom gromadzenia danych monitorowania dla kanału.

To pole nie ma znaczenia dla kanałów z typem *ChannelType* o wartości MQCHT_CLNTCONN.

Jest to jedna z następujących wartości:

- MQMON_OFF,
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

To jest pole wejściowe do wyjścia. Nie jest on obecny, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

ChannelName (MQCHAR20)

To pole określa nazwę definicji kanału.

Aby można było komunikować się z komputerem zdalnym, musi istnieć definicja kanału o tej samej nazwie na komputerze zdalnym.

W nazwie muszą być używane tylko znaki:

- Wielkie litery A-Z
- Małe litery a-z
- Cyfry 0-9
- Kropka (.)
- Prawy ukośnik (/)
- Podkreślenie (_)
- Znak procentu (%)

i być wyściełany do prawej z odstępami. Czołowe lub wewnętrzne odstępy nie są dozwolone.

Długość tego pola jest podana przez parametr MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)

To pole określa bieżący poziom gromadzenia danych statystycznych dla kanału.

To pole nie jest istotne dla kanałów z typem ChannelType z MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Jest to jedna z następujących wartości:

- MQMON_OFF,
- MQMON_LOW
- MQMON_MEDIUM
- MQMON_HIGH

To jest pole wejściowe do wyjścia. Nie jest on obecny, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

ChannelType (MQLONG)

To pole określa typ kanału.

Jest to jedna z następujących wartości:

MQCHT_SENDER

Nadawca.

SERWER_MQCHT_SERVER

Serwer.

MQCHT_RECEIVER

Odbiornik.

MQCHT_REQUESTER

Żądający.

MQCHT_CLNTCONN

Połączenie klienta.

MQCHT_SVRCONN

Serwer-połączenie (do użytku przez klientów).

MQCHT_CLUSSDR

Nadawca klastra.

MQCHT_CLUSRCVR

Odbiornik klastra.

ClientChannelWaga (MQLONG)

To pole określa wagę wpływającą na użycie definicji kanału połączenia klienckiego.

Atrybut wagi ClientChannel jest używany w taki sposób, że definicje kanałów klienta mogą być wybierane losowo na podstawie ich wagi, jeśli dostępna jest więcej niż jedna odpowiednia definicja. Gdy klient wysyła żądanie połączenia MQCONN do grupy menedżerów kolejek, określając nazwę menedżera kolejek rozpoczynającą się gwiazdką i w tabeli definicji kanału klienta (CCDT) jest dostępna więcej niż jedna odpowiednia definicja kanału, definicja do użycia jest wybierana losowo na podstawie wagi, z dowolnymi odpowiednimi definicjami wagi ClientChannel(0), które zostały wybrane jako pierwsze w kolejności alfabetycznej.

Określ wartość z zakresu od 0 do 99. Wartość domyślna to 0.

Wartość 0 wskazuje brak równoważenia obciążenia, a odpowiednie definicje są wybierane w porządku alfabetycznym. Aby włączyć równoważenie obciążenia, wybierz wartość z zakresu od 1 do 99, gdzie 1 to najniższa waga, a 99 to najwyższa waga. Rozkład komunikatów między dwoma lub większą liczbą kanałów z niezerowymi wagami jest proporcjonalny do stosunku tych współczynników korygujący. Na przykład trzy kanały z wartościami wagi ClientChannelWaga 2, 4 i 14 są wybierane w przybliżeniu 10%, 20% i 70% czasu. Ta dystrybucja nie jest gwarantowana.

Ten atrybut jest poprawny tylko dla typu kanału połączenia klienckiego.

To jest pole wejściowe do wyjścia. Pole nie jest obecne, jeśli *Wersja* jest mniejsza niż MQCD_VERSION_9.

ClusterPtr (MQPTR)

To pole służy do określania adresu w postaci listy nazw klastrów.

Jeśli wartość *ClustersDefined* jest większa od zera, adres ten jest adresem listy nazw klastrów. Kanał należy do każdego wymienionego klastra.

To pole ma zastosowanie tylko w przypadku kanałów z parametrem *ChannelType* MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

ClustersDefined (MQLONG)

To pole określa liczbę skupień, do których należy kanał.

To pole jest liczbą nazw klastrów wskazanych przez *ClusterPtr*. Wartość jest równa zero lub większa.

To pole ma zastosowanie tylko w przypadku kanałów z parametrem *ChannelType* MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)

To pole określa priorytet kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia wybiera miejsce docelowe o najwyższym priorytecie z zestawu miejsc docelowych wybranych w oparciu o pozycję w rankingu. Jeśli istnieją dwa możliwe docelowe menedżery kolejek, ten atrybut może zostać użyty do przełączenia awaryjnego jednego menedżera kolejek na inny menedżer kolejek. Wszystkie komunikaty są wysyłane do menedżera kolejek o najwyższym priorytecie do momentu zakończenia, a następnie komunikaty trafiają do menedżera kolejek przy użyciu kolejnego najwyższego priorytetu.

Wartość mieści się w zakresie od 0 do 9. Wartość domyślna to 0.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klastra menedżera kolejek](#).

CLWLChannelRank (MQLONG)

To pole określa klasyfikację kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia wybiera miejsce docelowe o najwyższej rangi. Jeśli ostatnim miejscem docelowym jest menedżer kolejek w innym klastrze, można ustawić rangę menedżerów kolejek pośrednich bramy (na przecięciu sąsiednich klastrów), tak aby algorytm wyboru poprawnie wybrał docelowy menedżer kolejek bliżej końcowego miejsca docelowego.

Wartość mieści się w zakresie od 0 do 9. Wartość domyślna to 0.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klastra menedżera kolejek](#).

CLWLChannelWeight (MQLONG)

To pole określa wagę kanału obciążenia klastra.

Waga kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia używa atrybutu "waga" kanału do przesunięcia wyboru miejsca docelowego, tak aby możliwe było wysyłanie większej liczby komunikatów do konkretnego komputera. Na przykład można nadać kanał na dużym serwerze UNIX większą "wagę" niż inny kanał na małym komputerze desktop PC, a algorytm wyboru wybiera serwer UNIX częściej niż komputer PC.

Wartość ta mieści się w zakresie od 1 do 99. Domyślną wartością jest 50.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klastra menedżera kolejek](#).

ConnectionAffinity (MQLONG)

To pole określa, czy aplikacje klienckie, które łączą wiele razy przy użyciu tej samej nazwy menedżera kolejek, używają tego samego kanału klienta.

Ten atrybut jest używany, jeśli dostępnych jest wiele definicji kanałów.

Wartość ta jest jedną z następujących wartości:

MQCAFTY_PREFEROWANE

Pierwsze połączenie w procesie odczytującej tabelę definicji kanału klienta (CCDT) tworzy listę odpowiednich definicji na podstawie wagi z odpowiednimi definicjami CLNTWGHT (0) jako pierwsza i w kolejności alfabetycznej. Każde połączenie w procesie próbuje nawiązać połączenie przy użyciu pierwszej definicji z listy. Jeśli nawiązanie połączenia nie powiedzie się, używana jest następna definicja. Definicje niepomysłnych definicji z wartościami CLNTWGHT innych niż 0 są przenoszone na koniec listy. Definicje CLNTWGHT(0) pozostają na początku listy i są wybierane w pierwszej kolejności przy każdym nawiązywaniu połączenia.

Każdy proces klienta z tą samą nazwą hosta zawsze tworzy tę samą listę.

W przypadku aplikacji klienckich napisanych w języku C, C++ lub środowisku programowania .NET (w tym w pełni zarządzanych .NET) lista jest aktualizowana, jeśli pakiet CCDT został zmodyfikowany od momentu utworzenia listy.

Ta wartość jest wartością domyślną.

MQCAFTY_NONE

Pierwsze połączenie w procesie odczytu CCDT tworzy listę odpowiednich definicji. Wszystkie połączenia w procesie wybierają odpowiednią definicję w oparciu o wagę każdej odpowiedniej definicji CLNTWGHT(0) wybranej najpierw zgodnie z porządkiem alfabetycznym.

W przypadku aplikacji klienckich napisanych w języku C, C++ lub środowisku programowania .NET (w tym w pełni zarządzanych .NET) lista jest aktualizowana, jeśli pakiet CCDT został zmodyfikowany od momentu utworzenia listy.

Ten atrybut jest poprawny tylko dla typu kanału połączenia klienckiego.

To jest pole wejściowe do wyjścia. Pole nie jest obecne, jeśli *Wersja* jest mniejsza niż MQCD_VERSION_9.

ConnectionName (MQCHAR264)

To pole określa nazwę połączenia dla kanału.

W przypadku kanałów odbiorczych klastra (jeśli jest określona) CONNAME odnosi się do lokalnego menedżera kolejek, a dla innych kanałów odnosi się do docelowego menedżera kolejek. Wartość określona przez użytkownika zależy od protokołu transmisji (*TransportType*), który ma być używany:

- W przypadku parametru MQXPT_LU62 jest to pełna nazwa partnerskiej jednostki logicznej.
- Dla MQXPT_NETBIOS jest to nazwa NetBIOS zdefiniowana na komputerze zdalnym.
- W przypadku MQXPT_TCP jest to albo nazwa hosta, adres sieciowy komputera zdalnego określonego w IPv4 w postaci dziesiętnej z kropkami lub IPv6 w formacie szesnastkowym, albo komputer lokalny dla kanałów odbierających klastry.
- W przypadku MQXPT_SPX jest to adres SPX składający się z 4-bajтового adresu sieciowego, 6-bajтового adresu węzła i dwubajтового numeru gniazda.

W przypadku definiowania kanału to pole nie jest istotne dla kanałów z *ChannelType* z MQCHT_SVRCONN lub MQCHT_RECEIVER. Jeśli jednak definicja kanału zostanie przekazana do wyjścia, pole to zawiera adres partnera, niezależnie od typu kanału.

Długość tego pola jest podana przez wartość MQ_CONN_NAME_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

DataConversion (MQLONG)

To pole określa, czy wysyłający agent kanału komunikatów próbuje przeprowadzić konwersję danych komunikatu aplikacji, jeśli odbierający agent kanału komunikatów nie może wykonać tej konwersji.

To pole ma zastosowanie tylko do komunikatów, które nie są segmentami komunikatów logicznych; agent MCA nigdy nie próbuje konwertować komunikatów, które są segmentami.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR. Jest to jedna z poniższych nazw:

MQCDC_SENDER_CONVERSION

Konwersja przez nadawcę.

MQCDC_NO_SENDER_CONVERSION

Brak konwersji przez nadawcę.

DefReconnect (MQLONG)

Atrybut kanału DefReconnect ustawia domyślną wartość atrybutu reconnection dla kanału połączenia klienta.

Domyślna opcja automatycznego ponownego nawiązywania połączenia z klientem. Klient IBM MQ MQI client można skonfigurować w taki sposób, aby automatycznie ponownie nawiązywał połączenie z aplikacją kliencką. Klient IBM MQ MQI client podejmuje próbę ponownego nawiązania połączenia z menedżerem kolejek po niepowodzeniu połączenia. Podejmowana jest próba ponownego nawiązania połączenia bez wysyłania wywołania MQI MQCONN lub MQCONNX przez klient aplikacji.

Ponowne połączenie jest opcją MQCONNX. Za pomocą atrybutu kanału DefReconnect można dodać zachowanie ponownego połączenia do istniejących aplikacji, które korzystają z produktu MQCONN. Istnieje również możliwość zmiany zachowania ponownego połączenia aplikacji, które korzystają z produktu MQCONNX.

Można również ustawić wartość DefRecon z pliku mqclient.ini, aby ustawić lub zmodyfikować zachowanie ponownego połączenia. Wartość DefRecon z pliku mqclient.ini ma pierwszeństwo przed atrybutem kanału DefReconnect.

Składnia

DefReconnect (MQRCN_NO | MQRCN_YES | MQRCN_Q_MGR | MQRCN_DISABLED)

Parametry

MQRCN_NO

MQRCN_NO to wartość domyślna.

O ile nie zostanie nadpisane przez produkt MQCONNX, klient nie jest ponownie połączony automatycznie.

MQRCN_YES

O ile nie zostanie nadpisane przez produkt MQCONNX, klient ponownie połączy się ponownie.

MQRCN_Q_MGR

O ile nie zostaną nadpisane przez produkt MQCONNX, klient ponownie łączy się ponownie, ale tylko do tego samego menedżera kolejek. Opcja QMGR ma taki sam efekt jak MQCNO_RECONNECT_Q_MGR.

MQRCN_DISABLED

Ponowne połączenie jest wyłączone, nawet jeśli jest wymagane przez program kliencki przy użyciu wywołania MQI produktu MQCONNX.

Klasy IBM MQ classes for Java nie obsługują automatycznego ponownego nawiązywania połączenia przez klient.

Tabela 192. Automatyczne ponowne połączenie zależy od wartości ustawionych w aplikacji i definicji kanału

DefReconnect	Opcje ponownego połączenia ustawione w aplikacji			
	MQCNO_RECONNE CT	MQCNO_RECONNE CT_Q_MGR	MQCNO_RECONNE CT_AS_DEF	MQCNO_RECONNE CT_DISABLED
MQRCN_NO	YES	QMGR	NO	NO
MQRCN_YES	YES	QMGR	YES	NO
MQRCN_Q_MGR	YES	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

Odsyłacze pokrewne

Opcje nawiązywania połączeń

Opcje, które sterują działaniem MQCONN.

Informacje pokrewne

[Automatyczne ponowne łączenie klienta](#)

[Ponowne połączenie kanału i klienta](#)

[Sekcja CHANNELS w pliku konfiguracyjnym klienta](#)

Opis (MQCHAR64)

To pole może być używane w komentarzach opisowych.

Treść tego pola nie ma znaczenia dla agentów kanałów komunikatów. Jednak może ona zawierać tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków o kodzie zero; jeśli jest to konieczne, jest dopełniany do prawej strony odstępami. W przypadku instalacji DBCS pole może zawierać znaki DBCS (z zastrzeżeniem maksymalnej długości pola 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zgodnie z definicją atrybutu menedżera kolejek produktu *CodedCharSetId*), te znaki mogą być tłumaczone niepoprawnie, jeśli to pole jest wysyłane do innego menedżera kolejek.

Długość tego pola jest podana przez parametr MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)

To pole określa maksymalny czas (w sekundach), przez jaki kanał oczekuje na dotarcie komunikatu do kolejki transmisji przed zakończeniem kanału.

Innymi słowy, określa interwał odłączania.

Wartość zero powoduje, że agent MCA czeka bezterminowo.

W przypadku kanałów połączenia z serwerem za pomocą protokołu TCP odstęp czasu reprezentuje wartość odłączania nieaktywności klienta określoną w sekundach. Jeśli połączenie z serwerem nie zostało odebrane przez klienta partnerskiego przez ten czas, to połączenie zostanie przerwane. Przedział czasu nieaktywności połączenia z serwerem ma zastosowanie tylko między wywołaniami interfejsu API produktu IBM MQ od klienta, więc żaden klient nie jest odłączony podczas długotrwałego wywołania MQGET z wywołaniem oczekiwania.

Ten atrybut nie ma zastosowania w przypadku kanałów połączenia z serwerem przy użyciu protokołów innych niż TCP.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR lub MQCHT_SVRCONN.

Długość ExitData(MQLONG)

To pole określa długość każdego elementu danych użytkownika w bajtach na liście elementów danych użytkownika wyjścia, które są adresowane przez pola *MsgUserDataPtr*, *SendUserDataPtr* i *ReceiveUserDataPtr*.

Ta długość nie musi być taka sama, jak wartość `MQ_EXIT_DATA_LENGTH`.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż `MQCD_VERSION_4`.

Długość ExitName(MQLONG)

To pole określa długość w bajtach każdej z nazw na listach nazw wyjść adresowanych przez pola *MsgExitPtr*, *SendExitPtr* i *ReceiveExitPtr*.

Ta długość nie musi być taka sama, jak wartość `MQ_EXIT_NAME_LENGTH`.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż `MQCD_VERSION_4`.

Lista HdrComp[2] (MQLONG)

To pole określa listę technik kompresji danych nagłówka, które są obsługiwane przez kanał.

Lista zawiera jedną lub więcej z następujących wartości:

MQCOMPRESS_NONE

Dane nagłówka nie są kompresowane.

MQCOMPRESS_SYSTEM

Dane nagłówka są kompresowane.

Nie używane wartości w tablicy są ustawione na wartość `MQCOMPRESS_NOT_AVAILABLE`.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż `MQCD_VERSION_8`.

HeartbeatInterval (MQLONG)

To pole określa czas (w sekundach) między przepływami pulsu.

Interpretacja tego pola zależy od typu kanału w następujący sposób:

- W przypadku typu kanału `MQCHT_SENDER`, `MQCHT_SERVER`, `MQCHT_RECEIVER` `MQCHT_REQUESTER`, `MQCHT_CLUSSDR` lub `MQCHT_CLUSRCVR` to pole to czas (w sekundach) między przepływami pulsu przekazywanemu z wysyłającego agenta MCA, gdy w kolejce transmisji nie ma żadnych komunikatów. Dzięki temu odbierający agent MCA ma możliwość wyciszenia kanału. Aby program *HeartbeatInterval* mógł być przydatny, musi być mniejszy niż *DiscInterval*.
- W przypadku typu kanału `MQCHT_CLNTCONN` lub `MQCHT_SVRCONN` z polem konwersacji współużytkowania `MQCD` ustawionym na wartość zero to pole to czas (w sekundach) między przepływami pulsu przekazywanych z agenta MCA serwera, gdy agent MCA wygenerował wywołanie `MQGET` z opcją `MQGMO_WAIT` w imieniu aplikacji klienckiej. Dzięki temu agent MCA serwera może obsługiwać sytuacje, w których połączenie klienta nie powiedzie się podczas operacji `MQGET` z `MQGMO_WAIT`.
- W przypadku typu kanału `MQCHT_CLNTCONN` lub `MQCHT_SVRCONN` z polem konwersacji współużytkowania `MQCD` ustawionym na wartość niezerową, to pole to czas (w sekundach) między przepływem pulsu, gdy nie są wysyłane lub odbierane przepływy danych. Dzięki temu kanał może być wydajnie wyciszony.

Wartość mieści się w zakresie od 0 do 999 999. Używana wartość jest większa z wartości określonych na stronie wysyłającej i odbierającej, chyba że po obu stronach zostanie określona wartość 0, w którym to przypadku nie występuje wymiana pulsu.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż `MQCD_VERSION_4`.

Przedział czasu KeepAlive(MQLONG)

To pole określa wartość przekazanej do stosu komunikacji dla czasu sprawdzania połączenia dla kanału.

Wartość ta jest stosowana w protokołach komunikacyjnych TCP/IP i SPX, chociaż nie wszystkie implementacje obsługują ten parametr.

Wartość mieści się w zakresie od 0 do 99 999; jednostki to sekundy. Wartość zero oznacza, że kanał keepalive nie jest włączony, mimo że funkcja keepalive może nadal występować, jeśli włączony jest protokół TCP/IP keepalive (a nie kanał keepalive). Poprawna jest również następująca wartość specjalna:

MQKAI_AUTO

Automatyczny.

Ta wartość wskazuje, że interwał sprawdzania połączenia jest obliczany na podstawie wynegocjowanego okresu pulsu w następujący sposób:

- Jeśli wynegocjowany przedział czasu pulsu jest większy od zera, przedział czasu sprawdzania połączenia jest interwał pulsu plus 60 sekund.
- Jeśli wynegocjowany przedział czasu pulsu wynosi zero, używany przedział czasu sprawdzania połączenia wynosi zero.
- W systemie z/OS przy użyciu protokołu TCP/IP, gdy w obiekcie menedżera kolejek jest określony parametr TCPKEEP (YES), występuje protokół TCP/IP.
- W innych środowiskach podtrzymywanie połączenia TCP/IP występuje, gdy parametr KEEPALIVE=YES jest określony w sekcji TCP w rozproszonym pliku konfiguracyjnym kolejkowania.

To pole jest istotne tylko dla kanałów, które mają *TransportType* z MQXPT_TCP lub MQXPT_SPX.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

LocalAddress (MQCHAR48)

To pole określa lokalny adres TCP/IP zdefiniowany dla kanału dla komunikacji wychodzącej.

To pole jest puste, jeśli dla komunikacji wychodzącej nie zdefiniowano konkretnego adresu. Adres może opcjonalnie zawierać numer portu lub zakres numerów portów. Format tego adresu jest następujący:

```
[ip-addr] [(low-port[,high-port])]
```

gdzie nawiasy kwadratowe ([]) oznaczają informacje opcjonalne, ip-addr jest określany w postaci IPv4 w postaci dziesiętnej z kropkami, IPv6 w postaci szesnastkowej lub alfanumerycznej, a low-port i high-port to numery portów ujęte w nawiasy. Wszystkie są opcjonalne.

Konkretny adres IP, port lub zakres portów dla komunikacji wychodzącej jest przydatny w scenariuszach odtwarzania, w których kanał jest restartowany na innym stosie TCP/IP.

Produkt *LocalAddress* jest podobny w postaci do produktu *ConnectionName*, ale nie może być z nim mylony. *LocalAddress* określa parametry komunikacji lokalnej, podczas gdy *ConnectionName* określa, jak dotrzeć do menedżera kolejek zdalnych.

To pole jest istotne tylko dla kanałów z *TransportType* MQXPT_TCP i *ChannelType* z MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Długość tego pola jest podana przez wartość MQ_LOCAL_ADDRESS_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)

To pole określa długość (w bajtach) pełnego identyfikatora użytkownika MCA wskazanego przez *LongMCAUserIdPtr*.

To pole nie ma znaczenia dla kanałów z *ChannelType* z MQCHT_CLNTCONN.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)

To pole określa adres identyfikatora użytkownika długiego MCA.

Jeśli wartość *LongMCAUserIdLength* jest większa od zera, to pole to jest adresem pełnego identyfikatora użytkownika MCA. Długość pełnego identyfikatora jest nadawana przez

LongMCAUserIdLength. Pierwsze 12 bajtów identyfikatora użytkownika MCA znajduje się również w polu *MCAUserIdentifier*.

Szczegółowe informacje na temat identyfikatora użytkownika MCA można znaleźć w opisie pola *MCAUserIdentifier*.

To pole nie ma znaczenia dla kanałów z *ChannelType* z MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN lub MQCHT_CLUSSDR.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongRemoteUserIdLength (MQLONG)

To pole określa długość (w bajtach) pełnego identyfikatora użytkownika zdalnego wskazanego przez *LongRemoteUserIdPtr*.

To pole jest istotne tylko dla kanałów z *ChannelType* z MQCHT_CLNTCONN lub MQCHT_SVRCONN.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)

To pole określa adres długiego zdalnego identyfikatora użytkownika.

Jeśli wartość *LongRemoteUserIdLength* jest większa od zera, oznacza to, że jest to adres pełnego identyfikatora użytkownika zdalnego. Długość pełnego identyfikatora jest nadawana przez *LongRemoteUserIdLength*. Pierwsze 12 bajtów identyfikatora zdalnego użytkownika znajduje się również w polu *RemoteUserIdentifier*.

Szczegółowe informacje na temat identyfikatora zdalnego użytkownika można znaleźć w opisie pola *RemoteUserIdentifier*.

To pole jest istotne tylko dla kanałów z *ChannelType* z MQCHT_CLNTCONN lub MQCHT_SVRCONN.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Licznik LongRetry(MQLONG)

To pole określa licznik używany po wyczerpaniu liczby określonej przez *ShortRetryCount*.

Określa ona maksymalną liczbę kolejnych prób nawiązania połączenia z komputerem zdalnym, w określonych odstępach czasu określonych przez program *LongRetryInterval*, przed wyrejestrowaniem błędu do operatora.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Odstęp czasu LongRetry(MQLONG)

To pole określa maksymalną liczbę sekund oczekiwania przed ponowną próbą nawiązania połączenia z komputerem zdalnym.

Odstęp czasu między ponownymi próbami może zostać przedłużony, jeśli kanał musi oczekiwać na aktywne działanie.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

MaxInstances (MQLONG)

To pole określa maksymalną liczbę jednoczesnych instancji pojedynczego kanału połączenia z serwerem, który może być uruchomiony.

To pole jest używane tylko w kanałach połączeń z serwerem.

Pole może mieć wartość z zakresu od 0 do 999 999 999. Wartość zero uniemożliwia dostęp do klienta.

Wartością domyślną tego pola jest 999 999 999.

Jeśli wartość tego pola zostanie zmniejszona do liczby, która jest mniejsza niż liczba instancji kanału połączenia z serwerem, które są obecnie uruchomione, te działające instancje nie będą miały wpływu na te instancje. Jednak nowe instancje nie mogą być uruchamiane, dopóki nie przestaną działać wystarczająca liczba instancji, tak aby liczba obecnie działających instancji była mniejsza niż wartość pola.

MaxInstancesPerClient (MQLONG)

To pole określa maksymalną liczbę jednoczesnych instancji pojedynczego kanału połączenia z serwerem, które mogą być uruchamiane z jednego klienta.

W tym kontekście połączenia, które pochodzą z tego samego adresu sieci zdalnej, są uznawane za pochodzące od tego samego klienta.

To pole jest używane tylko w kanałach połączeń z serwerem.

Pole może mieć wartość z zakresu od 0 do 999 999 999. Wartość zero uniemożliwia dostęp do klienta.

Wartością domyślną tego pola jest 999 999 999.

Jeśli wartość tego pola zostanie zmniejszona do liczby, która jest mniejsza niż liczba instancji kanału połączenia z serwerem, które są obecnie uruchomione przez poszczególne klienty, te działające instancje nie będą miały wpływu na te instancje. Jednak nowe instancje z dowolnego z tych klientów nie mogą zostać uruchomione, dopóki nie przestaną działać wystarczająca liczba istniejących instancji, tak aby liczba obecnie działających instancji, pochodzących od klienta próbującego rozpocząć nową, była mniejsza niż wartość pola.

MaxMsgDługość (MQLONG)

To pole określa maksymalną długość komunikatu, która może być przesyłana w kanale.

Jest ona porównywana z wartością kanału zdalnego i z tych dwóch wartości niższą wartością jest bieżąca wartość maksymalna.

MCAName (MQCHAR20)

To pole jest polem zastrzeżonym.

Wartość tego pola jest pusta.

Długość tego pola jest podana przez wartość MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)

To pole określa identyfikator zabezpieczeń dla agenta MCA.

To pole nie ma znaczenia dla kanałów z *ChannelType* z MQCHT_CLNTCONN.

Następująca wartość specjalna wskazuje, że nie ma identyfikatora zabezpieczeń:

MQSID_NONE

Nie określono identyfikatora zabezpieczeń.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQSID_NONE_ARRAY; ta stała ma taką samą wartość jak MQSID_NONE, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe/wyjściowe do wyjścia. Długość tego pola jest podana przez wartość MQ_SECURITY_ID_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

MCAType (MQLONG)

To pole określa typ programu agenta kanału komunikatów.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Wartość ta jest jedną z następujących wartości:

MQMCAT_PROCESS

proces.

Agent kanału komunikatów jest uruchamiany jako oddzielny proces.

MQMCAT_THREAD

Wątek (IBM i, UNIX i Windows).

Agent kanału komunikatów jest uruchamiany jako oddzielny wątek.

To pole nie jest obecne, gdy *wersja* jest mniejsza niż MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)

To pole określa identyfikator użytkownika dla agenta kanału komunikatów (MCA).

W tym polu używane są pierwsze 12 bajtów identyfikatora użytkownika MCA i mogą być ustawiane przez agenta zabezpieczeń.

Istnieją dwa pola, które zawierają identyfikator użytkownika MCA:

- *MCAUserIdentifier* zawiera pierwsze 12 bajtów identyfikatora użytkownika MCA, a jeśli identyfikator jest krótszy niż 12 bajtów, jest dopełniany odstępami. *MCAUserIdentifier* może być pusta.
- *LongMCAUserIdPtr* wskazuje na pełny identyfikator użytkownika MCA, który może być dłuższy niż 12 bajtów. Jego długość jest podawana przez produkt *LongMCAUserIdLength*. Pełny identyfikator nie zawiera odstępów końcowych i nie jest zakończony znakiem o kodzie zero. Jeśli identyfikator jest pusty, *LongMCAUserIdLength* ma wartość zero, a wartość *LongMCAUserIdPtr* jest niezdefiniowana.

Uwaga: *LongMCAUserIdPtr* nie jest obecny, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Jeśli identyfikator użytkownika MCA nie jest pusty, określa on identyfikator użytkownika, który ma być używany przez agenta kanału komunikatów do autoryzacji w celu uzyskania dostępu do zasobów produktu IBM MQ . W przypadku typów kanałów MQCHT_REQUESTER, MQCHT_RECEIVER i MQCHT_CLUSRCVR, jeśli PutAuthority ma wartość MQPA_DEFAULT, jest to identyfikator użytkownika używany do sprawdzania autoryzacji dla operacji umieszczania w kolejkach docelowych.

Jeśli identyfikator użytkownika MCA jest pusty, agent kanału komunikatów używa jego domyślnego identyfikatora użytkownika.

Identyfikator użytkownika MCA może być ustawiony przez wyjście zabezpieczeń, aby wskazać identyfikator użytkownika, który musi być używany przez agenta kanału komunikatów. Wyjście może zmienić wartość *MCAUserIdentifier* lub łańcuch wskazujący na *LongMCAUserIdPtr*. Jeśli oba elementy są zmieniane, ale różnią się od siebie, agent MCA używa produktu *LongMCAUserIdPtr* w preferencjach produktu *MCAUserIdentifier*. Jeśli wyjście zmieni długość łańcucha adresowanego przez *LongMCAUserIdPtr*, należy odpowiednio ustawić wartość *LongMCAUserIdLength* . Jeśli wyjście zwiększa długość identyfikatora, wyjście musi przydzielić pamięć o wymaganej długości, ustawić tę pamięć na wymagany identyfikator, a następnie umieścić adres tej pamięci w programie *LongMCAUserIdPtr*. Wyjście jest odpowiedzialne za zwalnianie tej pamięci, gdy wyjście zostanie później wywołane z powodu MQXR_TERM.

W przypadku kanałów z wartością *ChannelType* parametru MQCHT_SVRCONN, jeśli wartość *MCAUserIdentifier* w definicji kanału jest pusta, kopiowany jest do niego dowolny identyfikator użytkownika przestany z klienta. Ten identyfikator użytkownika (po dowolnej modyfikacji przez wyjście zabezpieczeń na serwerze) jest tym, w którym zakłada się, że aplikacja kliencka działa w ramach.

Identyfikator użytkownika agenta MCA nie ma znaczenia dla kanałów z *ChannelType* MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Jest to pole wejściowe/wyjściowe do wyjścia. Długość tego pola jest podana przez wartość `MQ_USER_ID_LENGTH`. To pole nie jest obecne, gdy wartość *Version* jest mniejsza niż `MQCD_VERSION_2`.

ModeName (MQCHAR8)

To pole określa nazwę trybu LU 6.2 .

To pole ma znaczenie tylko wtedy, gdy protokołem transmisji (*TransportType*) jest `MQXPT_LU62`, a *ChannelType* to nie jest `MQCHT_SVRCONN` ani `MQCHT_RECEIVER`.

To pole jest zawsze puste. Informacje te są zawarte w obiekcie po stronie komunikacyjnej.

Długość tego pola jest podana przez wartość `MQ_MODE_NAME_LENGTH`.

Lista MsgComp[16] (MQLONG)

To pole określa listę technik kompresji danych komunikatu, które są obsługiwane przez kanał.

Lista zawiera jedną lub więcej z następujących wartości:

MQCOMPRESS_NONE

Dane komunikatu nie są kompresowane.

MQCOMPRESS_RLE

Kompresja danych komunikatu jest wykonywana przy użyciu kodowania grupowego.

MQCOMPRESS_ZLIBFAST

Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowana jest szybka kompresja.

MQCOMPRESS_ZLIBHIGH

Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowany jest wysoki poziom kompresji.

Nie używane wartości w tablicy są ustawione na wartość `MQCOMPRESS_NOT_AVAILABLE`.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż `MQCD_VERSION_8`.

MsgExit (MQCHARn)

To pole określa nazwę wyjścia komunikatu kanału.

Jeśli ta nazwa jest niepusta, wyjście jest wywoływane w następujących godzinach:

- Natychmiast po pobraniu komunikatu z kolejki transmisji (nadawca lub serwer) lub bezpośrednio przed umieszczeniem komunikatu w kolejce docelowej (odbiorniku lub requesterze).

Wyjście otrzymuje cały komunikat aplikacji i nagłówek kolejki transmisji do modyfikacji.

- Przy inicjalizacji i zakończeniu kanału.

To pole nie ma znaczenia dla kanałów z *ChannelType* `MQCHT_SVRCONN` lub `MQCHT_CLNTCONN`; wyjście komunikatu nigdy nie jest wywoływane dla takich kanałów.

Informacje na temat zawartości tego pola w różnych środowiskach zawiera sekcja [“MQCD-definicja kanału” na stronie 1569](#) .

Długość tego pola jest podana przez wartość `MQ_EXIT_NAME_LENGTH`.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

MsgExitPtr (MQPTR)

To pole określa adres pierwszego pola *MsgExit* .

Jeśli wartość *MsgExitsDefined* jest większa od zera, adres ten jest adresem listy nazw każdego wyjścia komunikatu kanału w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełniona do prawej strony odstępami. Istnieją pola *MsgExitsDefined* , które sąsiadują ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, chociaż wyjście kanału komunikatów nie podejmuje żadnych jawnych działań-nie zmienia się, które wyjścia są wywoływane.

Jeśli parametr *MsgExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

MsgExitsZdefiniowane (MQLONG)

To pole określa liczbę wyjść komunikatów kanału zdefiniowanych w łańcuchu.

Wartość ta jest większa lub równa zero.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Liczba MsgRetry(MQLONG)

To pole określa, ile razy agent MCA próbuje umieścić komunikat, po pierwszej próbie.

To pole wskazuje, ile razy agent MCA próbuje wykonać operację otwarcia lub umieszczenia, jeśli pierwsza operacja MQOPEN lub MQPUT kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED. Wpływ tego atrybutu zależy od tego, czy parametr *MsgRetryExit* jest pusty, czy też nie jest pusty:

- Jeśli pole *MsgRetryExit* jest puste, atrybut *MsgRetryCount* określa, czy próby MCA będą ponawiać próby. Jeśli wartość atrybutu wynosi zero, próby nie są podejmowane. Jeśli wartość atrybutu jest większa od zera, próby są podejmowane w odstępach czasu podanych przez atrybut *MsgRetryInterval*.

Próby są podejmowane tylko dla następujących kodów przyczyny:

- MQRC_PAGESET_FULL
- MQRC_PUT_INHIBITED
- MQRC_Q_FULL

W przypadku innych kodów przyczyny, agent MCA przechodzi natychmiast do normalnego przetwarzania awarii, bez ponawiania błędnego komunikatu.

- Jeśli parametr *MsgRetryExit* jest niepusty, atrybut *MsgRetryCount* nie ma wpływu na agenta MCA. Zamiast tego jest to wyjście z ponowieniem komunikatu, które określa, ile razy próbowano wykonać ponowienie, oraz w jakich odstępach czasu. Wyjście jest wywoływane nawet wtedy, gdy atrybut *MsgRetryCount* ma wartość zero.

Atrybut *MsgRetryCount* jest dostępny dla wyjścia w strukturze MQCD, ale wyjście nie jest wymagane, aby uhonorować go-ponowne próby są kontynuowane w nieskończoność do momentu, aż wyjście zwróci MQXCC_SUPPRESS_FUNCTION w polu *ExitResponse* produktu MQCXP.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

To pole nie występuje, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

MsgRetryWyjście (MQCHARn)

To pole określa nazwę wyjścia dla ponowienia komunikatu kanału.

Wyjście ponowienia komunikatu jest to wyjście wywoływane przez agenta MCA, gdy agent MCA otrzyma kod zakończenia MQCC_FAILED z wywołania MQOPEN lub MQPUT. Celem wyjścia jest określenie odstępu czasu, przez który agent MCA oczekuje przed ponowną próbą wykonania operacji MQOPEN lub MQPUT. Alternatywnie, wyjście można ustawić, aby nie spróbować ponownie operacji.

Wyjście jest wywoływane dla wszystkich kodów przyczyny, dla których kod zakończenia MQCC_FAILED-ustawienia wyjścia określają kody przyczyny, które mają być ponawiane przez agenta MCA, liczby prób i w jakich odstępach czasu.

Jeśli operacja nie zostanie jeszcze podjęta, agent MCA wykonuje normalne przetwarzanie niepowodzenia. Przetwarzanie to obejmuje wygenerowanie komunikatu o wyjątku (jeśli jest określony przez nadawcę) i umieszczenie oryginalnego komunikatu w kolejce niedostarczonych komunikatów lub usunięcie komunikatu (zgodnie z tym, czy nadawca określił wartość MQRO_DEAD_LETTER_Q lub MQRO_DISCARD_MSG). Niepowodzenia związane z kolejką niedostarczonych komunikatów (na przykład pełna kolejka niedostarczonych komunikatów) nie powodują wywołania wyjścia dla ponowienia komunikatu.

Jeśli nazwa wyjścia jest niepusta, to wyjście jest wywoływane w następujących godzinach:

- Bezpośrednio przed wykonaniem oczekiwania przed ponowną próbą dostarczenia komunikatu
- Przy inicjowaniu i zakończeniu kanału

Informacje na temat zawartości tego pola w różnych środowiskach zawiera sekcja [“MQCD-definicja kanału”](#) na stronie 1569 .

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

Długość tego pola jest podana przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

To pole nie występuje, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

Przedział czasu MsgRetry(MQLONG)

To pole określa minimalny odstęp czasu (w milisekundach), po którym operacja otwierania lub umieszczania jest ponawiana.

Wpływ tego atrybutu zależy od tego, czy parametr *MsgRetryExit* jest pusty, czy też nie jest pusty:

- Jeśli pole *MsgRetryExit* jest puste, atrybut *MsgRetryInterval* określa minimalny okres, przez jaki agent MCA oczekuje przed ponowieniem komunikatu, jeśli pierwsze wywołanie MQOPEN lub MQPUT zakończy się niepowodzeniem z kodem zakończenia MQCC_FAILED. Wartość zero oznacza, że ponowienie zostanie wykonane tak szybko, jak to możliwe po poprzedniej próbie. Ponowne próby są wykonywane tylko wtedy, gdy wartość *MsgRetryCount* jest większa od zera.

Ten atrybut jest również używany jako czas oczekiwania, jeśli wyjście komunikatu-retry zwraca niepoprawną wartość w polu *MsgRetryInterval* w produkcie MQCXP.

- Jeśli parametr *MsgRetryExit* nie jest pusty, atrybut *MsgRetryInterval* nie ma wpływu na agenta MCA. Zamiast tego jest to wyjście z ponowieniem komunikatu, które określa czas oczekiwania agenta MCA. Atrybut *MsgRetryInterval* jest dostępny dla wyjścia w strukturze MQCD, ale wyjście nie jest wymagane, aby go uhonorować.

Wartość mieści się w zakresie od 0 do 999 999 999.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

To pole nie występuje, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)

To pole określa dane użytkownika wyjścia dla ponowienia komunikatu kanału.

Dane te są przekazywane do wyjścia komunikatu kanału-wyjście ponawiania w polu *ExitData* parametru *ChannelExitParms* (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji MCA wszystkie zmiany wprowadzone w treści tego pola przez wyjście dowolnego typu są zachowywane przez agenta MCA i są widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Takie zmiany nie wpływają na definicję kanału używaną przez inne instancje MCA. Mogą być używane dowolne znaki (w tym dane binarne).

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH. To pole nie występuje, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

To pole nie ma znaczenia w produkcie IBM MQ for IBM i.

Dane MsgUser(MQCHAR32)

To pole określa dane użytkownika wyjścia komunikatu kanału.

Dane te są przekazywane do wyjścia komunikatów kanału w polu *ExitData* w parametrze *ChannelExitParms* (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji MCA wszystkie zmiany wprowadzone w treści tego pola przez wyjście dowolnego typu są zachowywane przez agenta MCA i są widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Takie zmiany nie wpływają na definicję kanału używaną przez inne instancje MCA. Mogą być używane dowolne znaki (w tym dane binarne).

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma znaczenia w produkcie IBM MQ for IBM i.

MsgUserDataPtr (MQPTR)

To pole określa adres pierwszego pola *MsgUserData*.

Jeśli wartość *MsgExitsDefined* jest większa od zera, adres ten jest adresem listy elementów danych użytkownika dla każdego wyjścia komunikatu kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełniony do prawej strony odstępami. Istnieją pola *MsgExitsDefined*, które sąsiadują ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są ustawiane na wartości puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjścia, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane na wyjściu.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Dzięki temu jedno wyjście może przekazać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane na żadnych zmianach, na przykład w razie potrzeby w tych polach można zapisywać dane binarne.

Jeśli parametr *MsgExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

NetworkPriority (MQLONG)

To pole określa priorytet połączenia sieciowego dla kanału.

Jeśli dostępnych jest wiele ścieżek do określonego miejsca docelowego, wybierana jest ścieżka o najwyższym priorytecie. Wartość ta jest z zakresu od 0 do 9; 0 oznacza najniższy priorytet.

To pole ma zastosowanie tylko w przypadku kanałów z parametrem *ChannelType* MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)

To pole określa szybkość, z jaką nietrwałe komunikaty są przemieszczane przez kanał.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Wartość ta jest jedną z następujących wartości:

MQNPMS_NORMAL

Normalna prędkość.

Jeśli kanał jest zdefiniowany jako MQNPMS_NORMAL, komunikaty nietrwałe są przemieszczane przez kanał z normalną szybkością. Ma to zaletę, że komunikaty te nie zostaną utracone, jeśli wystąpi awaria kanału. Komunikaty trwałe i nietrwałe w tej samej kolejce transmisji zachowują swoje porządki względem siebie.

MQNPMS_FAST

Szybka prędkość.

Jeśli kanał jest zdefiniowany jako MQNPMS_FAST, komunikaty nietrwałe są przemieszczane przez kanał z szybkością szybkiego ruchu. Zwiększa to przepustowość kanału, ale oznacza, że komunikaty nietrwałe są tracone, jeśli wystąpi awaria kanału. Ponadto nietrwałe komunikaty mogą przeskoczyć przed trwałymi komunikatami, które oczekują w tej samej kolejce transmisji, co oznacza, że kolejność komunikatów nietrwałych nie jest obsługiwana względem trwałych komunikatów. Jednak kolejność komunikatów nietrwałych w stosunku do siebie jest zachowywana. Podobnie, kolejność komunikatów trwałych względem siebie jest zachowywana.

Hasło (MQCHAR12)

To pole określa hasło używane przez agenta kanału komunikatów podczas próby zainicjowania bezpiecznej sesji SNA z agentem zdalnego kanału komunikatów.

To pole może być niepuste tylko w systemach Systemy UNIXi Windowsi jest odpowiednie tylko dla kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER lub MQCHT_CLNTCONN. W systemie z/OS to pole nie jest istotne.

Długość tego pola jest podana przez wartość MQ_PASSWORD_LENGTH. Używane są jednak tylko pierwsze 10 znaków.

To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

PropertyControl (MQLONG)

To pole określa, co dzieje się z właściwościami komunikatów, gdy komunikat ma być wysyłany do menedżera kolejek w wersji V6 lub wcześniejszej (menedżer kolejek, który nie rozumie pojęcia deskryptora właściwości).

Możliwe wartości:

KOMPATYBILNA_MQPROP_KOMPATYBILNOŚCI

Jeśli komunikat zawiera właściwość z przedrostkiem **mcd.**, **jms.**, **usr.** lub **mqext.**, wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2. W przeciwnym razie wszystkie właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskrypcorze komunikatu (lub rozszerzeniu), są usuwane i nie są już dostępne dla aplikacji.

Ta wartość jest wartością domyślną. Pozwala ona aplikacjom, które oczekują, że właściwości związane z produktem JMS będą znajdować się w nagłówku MQRFH2 danych komunikatu, aby kontynuować pracę bez modyfikacji.

MQPROP_NONE

Wszystkie właściwości komunikatu, z wyjątkiem tych właściwości w deskrypcorze komunikatu (lub rozszerzeniu), są usuwane z komunikatu przed wysłaniem komunikatu do zdalnego menedżera kolejek.

MQPROP_ALL

Wszystkie właściwości komunikatu są dołączane do komunikatu, gdy jest on wysyłany do menedżera kolejek zdalnych. Właściwości te, z wyjątkiem tych, które znajdują się w deskrypcorze komunikatu (lub rozszerzeniu), zostają umieszczone w jednym lub większej liczbie nagłówków MQRFH2 danych komunikatu.

Ten atrybut ma zastosowanie do kanałów nadawcy, serwera, nadawcy klastra i odbiornika klastra.

“MQIA_ * (selektory atrybutów całkowitych)” na stronie 129

“MQPROP_ * (wartości sterujące właściwości kolejki i kanału oraz maksymalna długość właściwości)” na stronie 170

PutAuthority (MQLONG)

To pole określa, czy identyfikator użytkownika w informacjach kontekstowych powiązanych z komunikatem jest używany do ustanawiania uprawnień do umieszczenia komunikatu w kolejce docelowej.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR. Jest to jedna z poniższych nazw:

MQPA_DEFAULT

Używany jest domyślny identyfikator użytkownika.

MQPA_CONTEXT

Używany jest identyfikator użytkownika kontekstu.

MQPA_ALTERNATE_OR_MCA

Używany jest identyfikator użytkownika z pola UserIdentifier deskryptora komunikatu. Żaden ID użytkownika odebrany z sieci nie jest używany. Ta wartość jest obsługiwana tylko w systemie z/OS.

MQPA_ONLY_MCA,

Używany jest domyślny identyfikator użytkownika. Żaden ID użytkownika odebrany z sieci nie jest używany. Ta wartość jest obsługiwana tylko w systemie z/OS.

QMgrName (MQCHAR48)

To pole określa nazwę menedżera kolejek, z którym może nawiązać połączenie wyjście.

W przypadku kanałów z *ChannelType* innymi niż MQCHT_CLNTCONN to pole jest nazwą menedżera kolejek, z którym program zewnętrzny może nawiązać połączenie, który w systemach UNIX, Linux i Windows jest zawsze niepusty.

Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)

To pole określa nazwę wyjścia odbierania kanału.

Jeśli ta nazwa jest niepusta, wyjście jest wywoływane w następujących godzinach:

- Bezpośrednio przed przetworami odebranych danych sieciowych.

Wyjście jest nadawane kompletnym buforom transmisji, które zostały odebrane. Zawartość buforu może być modyfikowana zgodnie z wymaganiami.

- Przy inicjalizacji i zakończeniu kanału.

Informacje na temat zawartości tego pola w różnych środowiskach zawiera sekcja “MQCD-definicja kanału” na stronie 1569.

Długość tego pola jest podana przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

ReceiveExitPtr (MQPTR)

To pole określa adres pierwszego pola *ReceiveExit*.

Jeśli wartość *ReceiveExitsDefined* jest większa od zera, adres ten jest adresem listy nazw każdego kanału odbieranego przez kanał w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełniona do prawej strony odstępami. Istnieją pola *ReceiveExitsDefined*, które sąsiadują ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, chociaż wyjście kanału komunikatów nie podejmuje żadnych jawnych działań-nie zmienia się, które wyjścia są wywoływane.

Jeśli parametr *ReceiveExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

ReceiveExitsZdefiniowane (MQLONG)

To pole określa liczbę wyjść odbierania kanału zdefiniowanych w łańcuchu.

Wartość ta jest większa lub równa zero.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Dane ReceiveUser(MQCHAR32)

Ten kanał określa dane użytkownika wyjścia odbierania kanału.

Dane te są przekazywane do wyjścia odbierania kanału w polu *ExitData* w parametrze *ChannelExitParms* (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji MCA wszystkie zmiany wprowadzone w treści tego pola przez wyjście dowolnego typu są zachowywane przez agenta MCA i są widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Ma to zastosowanie do wyjść na różne rozmowy. Takie zmiany nie wpływają na definicję kanału używaną przez inne instancje MCA. Mogą być używane dowolne znaki (w tym dane binarne).

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma znaczenia w produkcie IBM MQ for IBM i.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)

To pole określa adres pierwszego pola *ReceiveUserData*.

Jeśli wartość *ReceiveExitsDefined* jest większa od zera, adres ten jest adresem listy elementów danych użytkownika dla każdego wyjścia odbierania kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełniony do prawej strony odstępami. Istnieją pola *ReceiveExitsDefined*, które sąsiadują ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są ustawiane na wartości puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjścia, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane na wyjściu.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Dzięki temu jedno wyjście może przekazać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane na żadnych zmianach, na przykład w razie potrzeby w tych polach można zapisywać dane binarne.

Jeśli parametr *ReceiveExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

RemotePassword (MQCHAR12)

To pole określa hasło partnera.

To pole zawiera poprawne informacje tylko wtedy, gdy parametr *ChannelType* ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_CLNTCONN hasło to jest hasłem, które zostało uzyskane ze środowiska. Wyjście może zostać wybrane w celu wystania go do wyjścia zabezpieczeń na serwerze.
- W przypadku wyjścia zabezpieczeń w kanale MQCHT_SVRCONN, pole to może zawierać hasło, które zostało uzyskane ze środowiska na kliencie, jeśli nie ma wyjścia zabezpieczeń klienta. Program obsługi wyjścia może użyć tego hasła, aby sprawdzić poprawność identyfikatora użytkownika w produkcie *RemoteUserIdentifier*.

Jeśli na kliencie znajduje się wyjście zabezpieczeń, informacje te można uzyskać w przepływie zabezpieczeń od klienta.

Długość tego pola jest podana przez wartość MQ_PASSWORD_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

RemoteSecurityId (MQBYTE40)

To pole określa identyfikator zabezpieczeń dla użytkownika zdalnego.

To pole jest istotne tylko dla kanałów z *ChannelType* z MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Następująca wartość specjalna wskazuje, że nie ma identyfikatora zabezpieczeń:

MQSID_NONE

Nie określono identyfikatora zabezpieczeń.

Wartość jest binarna zero dla długości pola.

W przypadku języka programowania C zdefiniowana jest również stała MQSID_NONE_ARRAY; ta stała ma taką samą wartość jak MQSID_NONE, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe do wyjścia. Długość tego pola jest podana przez wartość MQ_SECURITY_ID_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Następujące pola w tej strukturze nie są obecne, jeśli parametr *Version* jest mniejszy niż MQCD_VERSION_7.

Identyfikator RemoteUser (MQCHAR12)

To pole określa pierwsze 12 bajtów identyfikatora użytkownika od partnera.

Istnieją dwa pola, które zawierają identyfikator zdalnego użytkownika:

- *RemoteUserIdentifier* zawiera pierwsze 12 bajtów identyfikatora zdalnego użytkownika i jest dopełniane odstępami, jeśli identyfikator jest krótszy niż 12 bajtów. *RemoteUserIdentifier* może być pusta.
- *LongRemoteUserIdPtr* wskazuje na pełny identyfikator zdalnego użytkownika, który może być dłuższy niż 12 bajtów. Jego długość jest podawana przez produkt *LongRemoteUserIdLength*. Pełny identyfikator nie zawiera odstępów końcowych i nie jest zakończony znakiem o kodzie zero. Jeśli identyfikator jest pusty, *LongRemoteUserIdLength* ma wartość zero, a wartość *LongRemoteUserIdPtr* jest niezdefiniowana.

LongRemoteUserIdPtr nie jest obecny, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Identyfikator zdalnego użytkownika jest odpowiedni tylko dla kanałów z *ChannelType* MQCHT_CLNTCONN lub MQCHT_SVRCONN.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_CLNTCONN ta wartość jest identyfikatorem użytkownika, który został uzyskany z środowiska. Wyjście może zostać wybrane w celu wystania go do wyjścia zabezpieczeń na serwerze.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_SVRCONN to pole może zawierać identyfikator użytkownika, który został uzyskany ze środowiska na kliencie, jeśli nie ma wyjścia zabezpieczeń klienta. Program obsługi wyjścia może sprawdzić poprawność tego identyfikatora użytkownika (być może z hasłem w programie *RemotePassword*) i zaktualizować wartość w programie *MCAUserIdentifier*.

Jeśli na kliencie znajduje się wyjście zabezpieczeń, informacje te można uzyskać w przepływie zabezpieczeń od klienta.

Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

SecurityExit (MQCHARn)

To pole określa nazwę wyjścia zabezpieczeń kanału.

Jeśli ta nazwa jest niepusta, wyjście jest wywoływane w następujących godzinach:

- Natychmiast po uruchomieniu kanału.

Przed przesłaniem jakiegokolwiek komunikatu wyjście ma możliwość sprawdzenia przepływów zabezpieczeń w celu sprawdzenia poprawności autoryzacji połączenia.

- Po odebraniu odpowiedzi na przepływ komunikatów zabezpieczeń.

Wszystkie przepływy komunikatów bezpieczeństwa odebrane od procesora zdalnego na komputerze zdalnym są nadawane do wyjścia.

- Przy inicjalizacji i zakończeniu kanału.

Informacje na temat zawartości tego pola w różnych środowiskach zawiera sekcja [“MQCD-definicja kanału”](#) na stronie 1569.

Długość tego pola jest podana przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

Dane SecurityUser(MQCHAR32)

Ten kanał określa dane użytkownika wyjścia zabezpieczeń kanału.

Dane te są przekazywane do wyjścia zabezpieczeń kanału w polu *ExitData* parametru *ChannelExitParms* (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji MCA wszystkie zmiany wprowadzone w treści tego pola przez wyjście dowolnego typu są zachowywane przez agenta MCA i są widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Ma to zastosowanie do wyjść na różne rozmowy. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Mogą być używane dowolne znaki (w tym dane binarne).

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma znaczenia w produkcie IBM MQ for IBM i.

SendExit (MQCHARn)

To pole określa nazwę wyjścia wysyłania kanału.

Jeśli ta nazwa jest niepusta, wyjście jest wywoływane w następujących godzinach:

- Bezpośrednio przed wysłaniem danych w sieci.

Wyjście jest nadawane kompletnym buforom transmisji przed przesłaniem. Zawartość buforu może być modyfikowana zgodnie z wymaganiami.

- Przy inicjalizacji i zakończeniu kanału.

Informacje na temat zawartości tego pola w różnych środowiskach zawiera sekcja [“MQCD-definicja kanału”](#) na stronie 1569.

Długość tego pola jest podana przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

SendExitPtr (MQPTR)

To pole określa adres pierwszego pola *SendExit* .

Jeśli wartość *SendExitsDefined* jest większa od zera, adres ten jest adresem listy nazw każdego kanału wysyłający wyjście w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełniona do prawej strony odstępami. Istnieją pola *SendExitsDefined* , które sąsiadują ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, mimo że wyjście komunikatu nie podejmuje żadnych jawnych działań-nie zmienia się, które wyjścia są wywoływane.

Jeśli parametr *SendExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

SendExitsZdefiniowana (MQLONG)

To pole określa liczbę wyjść wysyłania kanału zdefiniowanych w łańcuchu.

Wartość ta jest większa lub równa zero.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Dane SendUser(MQCHAR32)

To pole określa, że kanał wysyła dane użytkownika wyjścia.

Dane te są przekazywane do wyjścia wysyłania kanału w polu *ExitData* w parametrze *ChannelExitParms* (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji MCA wszystkie zmiany wprowadzone w treści tego pola przez wyjście dowolnego typu są zachowywane przez agenta MCA i są widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Ma to zastosowanie do wyjść na różne rozmowy. Takie zmiany nie wpływają na definicję kanału używaną przez inne instancje MCA. Mogą być używane dowolne znaki (w tym dane binarne).

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma znaczenia w produkcie IBM MQ for IBM i.

SendUserDataPtr (MQPTR)

To pole określa adres pola *SendUserData* .

Jeśli wartość *SendExitsDefined* jest większa od zera, adres ten jest adresem listy elementów danych użytkownika dla każdego wyjścia komunikatu kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełniony do prawej strony odstępami. Istnieją pola *MsgExitsDefined* , które sąsiadują ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są ustawiane na wartości puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjścia, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane na wyjściu.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Dzięki temu jedno wyjście może przekazać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane na żadnych zmianach, na przykład w razie potrzeby w tych polach można zapisywać dane binarne.

Jeśli parametr *SendExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

W przypadku platform, w których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

SeqNumberWrap (MQLONG)

To pole określa najwyższy dopuszczalny numer kolejny komunikatu.

Po osiągnięciu tej wartości numery kolejne są zawijane w celu ponownego uruchomienia o 1.

Ta wartość jest niezbywalna i musi być zgodna zarówno w definicjach kanałów lokalnych, jak i zdalnych.

To pole nie ma znaczenia dla kanałów z *ChannelType* z MQCHT_SVRCONN lub MQCHT_CLNTCONN.

SharingConversations (MQLONG)

To pole określa maksymalną liczbę konwersacji, które mogą współużytkować instancję kanału powiązaną z tym kanałem.

To pole jest używane w połączeniu z klientem i kanałami połączeń serwera.

Wartość 0 oznacza, że kanał działa tak samo, jak w wersjach wcześniejszych niż IBM WebSphere MQ 7.0 w odniesieniu do następujących atrybutów:

- Współużytkowanie konwersacji
- Odczyt z wyprzedzeniem
- STOP CHANNEL(<channelname>) MODE(QUIESCE)
- Pulsowanie
- Asynchroniczne wykorzystanie klienta

Wartość 1 jest wartością minimalną dla zachowania produktu IBM MQ V7.0 . Chociaż dozwolona jest tylko jedna konwersacja dla instancji kanału, odczyt z wyprzedzeniem, wykorzystanie asynchroniczne oraz zachowanie programu CLNTCONN-SVRCONN w wersji 7, a także zatrzymywanie kanału wygaszania i zatrzymywanie kanału.

To jest pole wejściowe do wyjścia. Nie jest on obecny, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_9.

Wartością domyślną tego pola jest 10.

Uwaga: Limity *MaxInstances* i *MaxInstancesPerClient* zastosowane do kanału ograniczają liczbę instancji kanału, a nie liczbę konwersacji, które mogą być współużytkowane przez te instancje.

ShortConnectionNazwa (MQCHAR20)

To pole określa pierwsze 20 bajtów nazwy połączenia.

Jeśli pole *Version* ma wartość MQCD_VERSION_1, *ShortConnectionName* zawiera pełną nazwę połączenia.

Jeśli pole *Version* ma wartość MQCD_VERSION_2 lub większe, *ShortConnectionName* zawiera pierwsze 20 znaków nazwy połączenia. The full connection name is given by the *ConnectionName* field; *ShortConnectionName* and the first 20 characters of *ConnectionName* are identical.

Szczegółowe informacje na temat zawartości tego pola można znaleźć w sekcji *ConnectionName* .

Uwaga: Nazwa tego pola została zmieniona dla MQCD_VERSION_2 i kolejnych wersji zmaterializowanych tabel zapytań (MQCD); pole to było wcześniej nazywane *ConnectionName*.

Długość tego pola jest podana przez wartość MQ_SHORT_CONN_NAME_LENGTH.

Liczba ShortRetry(MQLONG)

To pole określa maksymalną liczbę prób nawiązania połączenia z komputerem zdalnym.

To pole jest maksymalną liczbą prób nawiązania połączenia z komputerem zdalnym, w określonych odstępach czasu określonych przez *ShortRetryInterval*, przed zużytym (zwykle dłuższym) *LongRetryCount* i *LongRetryInterval* .

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

ShortRetryprzedział czasu (MQLONG)

To pole określa maksymalną liczbę sekund oczekiwania przed ponowną próbą nawiązania połączenia z komputerem zdalnym.

Odstęp czasu między ponownymi próbami może zostać wydłużony, jeśli kanał musi oczekiwać na aktywne działanie.

To pole ma zastosowanie tylko w przypadku kanałów z *ChannelType* MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

SSLCipherSpec (MQCHAR32)

To pole określa specyfikację szyfru, która jest używana w przypadku korzystania z protokołu SSL.

Jeśli właściwość SSLCipherSpec jest pusta, kanał nie używa protokołu SSL. Jeśli pole to nie jest puste, to pole zawiera łańcuch określający atrybut CipherSpec w użyciu.

Ten parametr jest poprawny dla wszystkich typów kanałów. Jest on obsługiwany w systemach AIX, HP-UX, Linux, IBM i, Solaris, Windows z/OS. Jest on poprawny tylko dla typów kanałów typu transportu (TRPTYPE) TCP.

To jest pole wejściowe do wyjścia. Długość tego pola jest podana przez wartość MQ_SSL_CIPHER_SPEC_LENGTH. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

SSLClientAuth (MQLONG)

To pole określa, czy wymagane jest uwierzytelnianie klienta SSL.

To pole ma znaczenie tylko dla definicji kanału SVRCONN.

Jest to jedna z następujących wartości:

MQSCA_REQUIRED

Wymagane jest uwierzytelnianie klienta.

MQSCA_OPTIONAL

Uwierzytelnianie klienta jest opcjonalne.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

Długość parametru SSLPeerName(MQLONG)

To pole określa długość (w bajtach) nazwy węzła sieci SSL wskazywanej przez *SSLPeerNamePtr*.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)

To pole określa adres nazwy węzła sieci SSL.

Gdy certyfikat zostanie odebrany podczas pomyślnego uzgadniania SSL, nazwa wyróżniająca podmiotu certyfikatu jest kopiowana do pola MQCD, do którego dostęp jest uzyskiwany przez parametr SSLPeerNamePtr na końcu kanału, który odbiera certyfikat. Nadpisuje ona wartość parametru SSLPeerName dla kanału, jeśli ta wartość jest obecna w definicji kanału użytkownika lokalnego. Jeśli wyjście zabezpieczeń jest określone na tym końcu kanału, otrzymuje on nazwę wyróżniającą z certyfikatu równorzędnego na zmaterializowanych tabelach MQCD.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

Uwaga: Aplikacje wyjścia zabezpieczeń utworzone przed wydaniem produktu IBM MQ v7.1 mogą wymagać aktualizacji. Więcej informacji na ten temat zawiera sekcja [Programy obsługi wyjścia zabezpieczeń kanału](#).

StrucLength (MQLONG)

To pole określa długość (w bajtach) struktury MQCD.

Długość nie obejmuje żadnego z łańcuchów adresowanych przez pola wskaźnika znajdujące się w strukturze. Wartość ta jest jedną z następujących wartości:

MQCD_LENGTH_4

Długość struktury definicji kanału version-4 .

MQCD_LENGTH_5

Długość struktury definicji kanału version-5 .

MQCD_LENGTH_6

Długość struktury definicji kanału version-6 .

MQCD_LENGTH_7

Długość struktury definicji kanału version-7 .

MQCD_LENGTH_8

Długość struktury definicji kanału version-8 .

MQCD_LENGTH_9

Długość struktury definicji kanału version-9 .

Następująca stała określa długość bieżącej wersji:

MQCD_CURRENT_LENGTH

Długość bieżącej wersji struktury definicji kanału.

Uwaga: Te stałe mają wartości, które są specyficzne dla środowiska.

To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

TpName (MQCHAR64)

To pole określa nazwę programu transakcyjnego LU 6.2 .

To pole ma znaczenie tylko wtedy, gdy protokołem transmisji (*TransportType*) jest MQXPT_LU62, a *ChannelType* to nie jest MQCHT_SVRCONN ani MQCHT_RECEIVER.

To pole jest zawsze puste na platformach, na których informacje są zawarte w obiekcie komunikacji po stronie komunikacyjnej.

Długość tego pola jest podana przez wartość MQ_TP_NAME_LENGTH.

TransportType (MQLONG)

To pole określa protokół transmisji, który ma być używany.

Wartość nie jest sprawdzana, jeśli kanał został zainicjowany z drugiego końca.

Jest to jedna z następujących wartości:

MQXPT_LU62

Protokół transportowy LU 6.2 .

TCP MQXPT_TCP

Protokół transportowy TCP/IP.

MQXPT_NETBIOS

Protokół transportowy NetBIOS .

Ta wartość jest obsługiwana w następujących środowiskach: Windows.

MQXPT_SPX

Protokół transportowy SPX.

Ta wartość jest obsługiwana w następujących środowiskach: Windows oraz klienci IBM MQ połączone z tymi systemami.

UseDLQ (MQLONG)

To pole określa, czy kolejka niedostarczonych komunikatów (lub niedostarczona kolejka komunikatów) jest używana, gdy komunikaty nie mogą być dostarczane przez kanały.

Może zawierać jedną z następujących wartości:

MQUSEDLQ_NO

Komunikaty, które nie mogą być dostarczone przez kanał, są traktowane jako niepowodzenie. Kanał usuwa komunikat lub kanał kończy się, zgodnie z ustawieniem NPMSPEED.

MQUSEDLQ_YES

Jeśli atrybut menedżera kolejek DEADQ zawiera nazwę kolejki niedostarczonych komunikatów, to jest ona używana, w przeciwnym razie zachowanie jest takie samo jak dla NO. Wartość YES jest wartością domyślną.

UserIdentifier (MQCHAR12)

To pole określa identyfikator użytkownika używany przez agenta kanału komunikatów podczas próby zainicjowania bezpiecznej sesji SNA za pomocą zdalnego agenta kanału komunikatów.

To pole może być niepuste tylko w systemach Systemy UNIX i Windowsi jest odpowiednie tylko dla kanałów z *Channel Type* MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER lub MQCHT_CLNTCONN. W systemie z/OS to pole nie jest istotne.

Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH. Używane są jednak tylko pierwsze 10 znaków.

To pole nie jest obecne, gdy wartość *Version* jest mniejsza niż MQCD_VERSION_2.

Wersja (MQLONG)

Pole *Version* określa najwyższy numer wersji, który można ustawić dla struktury.

Wartość zależy od środowiska:

MQCD _VERSION_1

Struktura definicji kanału w wersji 1.

MQCD _VERSION_2

Struktura definicji kanału w wersji 2.

MQCD _VERSION_3

Struktura definicji kanału w wersji 3.

MQCD _VERSION_4

Struktura definicji kanału w wersji 4.

MQCD _VERSION_5

Struktura definicji kanału w wersji 5.

MQCD _VERSION_6

Wersja 6 struktura definicji kanału.

MQCD _VERSION_7

Struktura definicji kanału w wersji 7.

MQCD _VERSION_8

Struktura definicji kanału w wersji 8.

MQCD _VERSION_9

Struktura definicji kanału w wersji 9.

Wersja 9 jest najwyższą wartością, którą można ustawić na serwerze IBM WebSphere MQ 7.0 i IBM WebSphere MQ 7.0.1 na wszystkich platformach.

MQCD _VERSION_10

Struktura definicji kanału w wersji 10.

Wersja 10 jest najwyższą wartością, którą można ustawić na serwerze IBM WebSphere MQ 7.1 i IBM WebSphere MQ 7.5 na wszystkich platformach.

MQCD _VERSION_11

Struktura definicji kanału w wersji 11.

Wersja 11 jest najwyższą wartością, którą można ustawić na serwerze IBM MQ 8.0 na wszystkich platformach.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej wersji:

MQCD_CURRENT_VERSION

Wartość ustawiona w MQCD_CURRENT_VERSION jest bieżącą wersją używanej struktury definicji kanału.

Wartość MQCD_CURRENT_VERSION zależy od środowiska. Zawiera ona najwyższą wartość obsługiwaną przez platformę.

Produkt MQCD_CURRENT_VERSION nie jest używany do inicjowania domyślnych struktur podanych w nagłówku, kopii i dołączania plików udostępnionych dla różnych języków programowania. Domyślna inicjalizacja produktu Version zależy od platformy i wydania.

W przypadku produktu IBM WebSphere MQ 7.0 i nowszych wersji deklaracje MQCD w nagłówkach, kopiach i plikach włączanych są inicjowane do produktu MQCD_VERSION_6. Aby użyć dodatkowych pól MQCD, aplikacje muszą ustawić numer wersji na MQCD_CURRENT_VERSION. W przypadku pisania aplikacji, która jest przenośna między kilkoma środowiskami, należy wybrać wersję, która jest obsługiwana we wszystkich środowiskach.

Wskazówka: Gdy zostanie wprowadzona nowa wersja struktury MQCD, układ istniejącej części nie zostanie zmieniony. Wyjście musi sprawdzić numer wersji. Musi być ona równa lub większa od najniższej wersji, która zawiera pola, które muszą być używane przez wyjście.

XmitQName (MQCHAR48)

W tym polu podaje się nazwę kolejki transmisji, z której pobierane są komunikaty.

To pole ma zastosowanie tylko w przypadku kanałów z parametrem *ChannelType* o wartości MQCHT_SENDER lub MQCHT_SERVER.

Długość tego pola jest podana przez wartość MQ_Q_NAME_LENGTH.

Deklaracja C

Ta deklaracja jest deklaracją języka C dla struktury MQCD.

```
typedef struct tagMQCD MQCD;
typedef MQCD MQPOINTER PMQCD;
typedef PMQCD MQPOINTER PPMQCD;

struct tagMQCD {
    MQCHAR    ChannelName[20];          /* Channel definition name */
    MQLONG    Version;                  /* Structure version number */
    MQLONG    ChannelType;              /* Channel type */
    MQLONG    TransportType;            /* Transport type */
    MQCHAR    Desc[64];                 /* Channel description */
    MQCHAR    QMgrName[48];             /* Queue-manager name */
    MQCHAR    XmitQName[48];            /* Transmission queue name */
    MQCHAR    ShortConnectionName[20];  /* First 20 bytes of */
                                          /* connection name */
    MQCHAR    MCAName[20];              /* Reserved */
    MQCHAR    ModeName[8];              /* LU 6.2 Mode name */
    MQCHAR    TpName[64];               /* LU 6.2 transaction program */
                                          /* name */
    MQLONG    BatchSize;                /* Batch size */
    MQLONG    DiscInterval;             /* Disconnect interval */
    MQLONG    ShortRetryCount;          /* Short retry count */
    MQLONG    ShortRetryInterval;       /* Short retry wait interval */
    MQLONG    LongRetryCount;           /* Long retry count */
    MQLONG    LongRetryInterval;        /* Long retry wait interval */
    MQCHAR    SecurityExit[128];        /* Channel security exit name */
    MQCHAR    MsgExit[128];             /* Channel message exit name */
    MQCHAR    SendExit[128];            /* Channel send exit name */
    MQCHAR    ReceiveExit[128];         /* Channel receive exit name */
    MQLONG    SeqNumberWrap;            /* Highest allowable message */
}
```

```

MQLONG      MaxMsgLength;          /* sequence number */
MQLONG      PutAuthority;          /* Maximum message length */
MQLONG      DataConversion;        /* Put authority */
MQCHAR      SecurityUserData[32];  /* Data conversion */
                                          /* 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 */
                                          /* Password from partner */
MQCHAR      RemotePassword[12];
/* Ver:2 */
MQCHAR      MsgRetryExit[128];      /* Channel message retry exit */
                                          /* name */
MQCHAR      MsgRetryUserData[32];    /* Channel message retry exit */
                                          /* user data */
MQLONG      MsgRetryCount;          /* Number of times MCA will */
                                          /* try to put the message, */
                                          /* after first attempt has */
                                          /* failed */
MQLONG      MsgRetryInterval;        /* Minimum interval in */
                                          /* milliseconds after which */
                                          /* the open or put operation */
                                          /* will be retried */

/* Ver:3 */
MQLONG      HeartbeatInterval;      /* Time in seconds between */
                                          /* heartbeat flows */
MQLONG      BatchInterval;          /* Batch duration */
MQLONG      NonPersistentMsgSpeed;  /* Speed at which */
                                          /* nonpersistent messages are */
                                          /* sent */
MQLONG      StrucLength;            /* Length of MQCD structure */
MQLONG      ExitNameLength;         /* Length of exit name */
MQLONG      ExitDataLength;         /* Length of exit user data */
MQLONG      MsgExitsDefined;        /* Number of message exits */
                                          /* defined */
MQLONG      SendExitsDefined;        /* Number of send exits */
                                          /* defined */
MQLONG      ReceiveExitsDefined;     /* Number of receive exits */
                                          /* defined */
MQPTR       MsgExitPtr;             /* Address of first MsgExit */
                                          /* field */
MQPTR       MsgUserDataPtr;         /* Address of first */
                                          /* MsgUserData field */
MQPTR       SendExitPtr;            /* Address of first SendExit */
                                          /* field */
MQPTR       SendUserDataPtr;        /* Address of first */
                                          /* SendUserData field */
MQPTR       ReceiveExitPtr;         /* Address of first */
                                          /* ReceiveExit field */
MQPTR       ReceiveUserDataPtr;     /* Address of first */
                                          /* ReceiveUserData field */

/* Ver:4 */
MQPTR       ClusterPtr;             /* Address of a list of */
                                          /* cluster names */
MQLONG      ClustersDefined;        /* Number of clusters to */
                                          /* which the channel belongs */
MQLONG      NetworkPriority;        /* Network priority */

/* Ver:5 */
MQLONG      LongMCAUserIdLength;    /* Length of long MCA user */
                                          /* identifier */
MQLONG      LongRemoteUserIdLength; /* Length of long remote user */
                                          /* identifier */
MQPTR       LongMCAUserIdPtr;       /* Address of long MCA user */
                                          /* identifier */
MQPTR       LongRemoteUserIdPtr;    /* Address of long remote */
                                          /* user identifier */
MQBYTE40    MCASecurityId;          /* MCA security identifier */
MQBYTE40    RemoteSecurityId;       /* Remote security identifier */

/* Ver:6 */
MQCHAR      SSLCipherSpec[32];      /* SSL CipherSpec */

```

```

MQPTR      SSLPeerNamePtr;          /* Address of SSL peer name */
MQLONG     SSLPeerNameLength;       /* Length of SSL peer name */
MQLONG     SSLClientAuth;           /* Whether SSL client */
                                           /* authentication is required */
MQLONG     KeepAliveInterval;       /* Keepalive interval */
MQCHAR     LocalAddress[48];        /* Local communications */
                                           /* address */
MQLONG     BatchHeartbeat;          /* Batch heartbeat interval */
/* Ver:7 */
MQLONG     HdrCompList[2];          /* Header data compression */
                                           /* list */
MQLONG     MsgCompList[16];         /* Message data compression */
                                           /* list */
MQLONG     CLWLChannelRank;         /* Channel rank */
MQLONG     CLWLChannelPriority;      /* Channel priority */
MQLONG     CLWLChannelWeight;       /* Channel weight */
MQLONG     ChannelMonitoring;       /* Channel monitoring */
MQLONG     ChannelStatistics;       /* Channel statistics */
/* Ver:8 */
MQLONG     SharingConversations;    /* Limit on sharing */
                                           /* conversations */
MQLONG     PropertyControl;         /* Message property control */
MQLONG     MaxInstances;            /* Limit on SVRCONN channel */
                                           /* instances */
MQLONG     MaxInstancesPerClient;   /* Limit on SVRCONN channel */
                                           /* instances per client */
MQLONG     ClientChannelWeight;     /* Client channel weight */
MQLONG     ConnectionAffinity;      /* Connection affinity */
/* Ver:9 */
MQLONG     BatchDataLimit;          /* Batch data limit */
MQLONG     UseDLQ;                 /* Use Dead Letter Queue */
MQLONG     DefReconnect;            /* Default client reconnect */
                                           /* option */
/* Ver:10 */
MQCHAR64   CertificateLabel;        /* Certificate label */
/* Ver:11 */
};

```

Deklaracja języka COBOL

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCD.

```

** MQCD structure
  10 MQCD.
    ** Channel definition name
      15 MQCD-CHANNELNAME PIC X(20).
    ** Structure version number
      15 MQCD-VERSION PIC S9(9) BINARY.
    ** Channel type
      15 MQCD-CHANNELTYPE PIC S9(9) BINARY.
    ** Transport type
      15 MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
    ** Channel description
      15 MQCD-DESC PIC X(64).
    ** Queue-manager name
      15 MQCD-QMGRNAME PIC X(48).
    ** Transmission queue name
      15 MQCD-XMITQNAME PIC X(48).
    ** First 20 bytes of connection name
      15 MQCD-SHORTCONNECTIONNAME PIC X(20).
    ** Reserved
      15 MQCD-MCANAME PIC X(20).
    ** LU 6.2 Mode name
      15 MQCD-MODENAME PIC X(8).
    ** LU 6.2 transaction program name
      15 MQCD-TPNAME PIC X(64).
    ** Batch size
      15 MQCD-BATCHSIZE PIC S9(9) BINARY.
    ** Disconnect interval
      15 MQCD-DISCINTERVAL PIC S9(9) BINARY.
    ** Short retry count
      15 MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
    ** Short retry wait interval
      15 MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
    ** Long retry count
      15 MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
    ** Long retry wait interval
      15 MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
    ** Channel security exit name

```

```

15 MQCD-SECURITYEXIT PIC X(20).
** Channel message exit name
15 MQCD-MSGEXIT PIC X(20).
** Channel send exit name
15 MQCD-SENDEXIT PIC X(20).
** Channel receive exit name
15 MQCD-RECEIVEEXIT PIC X(20).
** Highest allowable message sequence number
15 MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
** Maximum message length
15 MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
** Put authority
15 MQCD-PUTAUTHORITY PIC S9(9) BINARY.
** Data conversion
15 MQCD-DATACONVERSION PIC S9(9) BINARY.
** Channel security exit user data
15 MQCD-SECURITYUSERDATA PIC X(32).
** Channel message exit user data
15 MQCD-MSGUSERDATA PIC X(32).
** Channel send exit user data
15 MQCD-SENDUSERDATA PIC X(32).
** Channel receive exit user data
15 MQCD-RECEIVEUSERDATA PIC X(32).
** Ver:1 **
** User identifier
15 MQCD-USERIDENTIFIER PIC X(12).
** Password
15 MQCD-PASSWORD PIC X(12).
** First 12 bytes of MCA user identifier
15 MQCD-MCAUSERIDENTIFIER PIC X(12).
** Message channel agent type
15 MQCD-MCATYPE PIC S9(9) BINARY.
** Connection name
15 MQCD-CONNECTIONNAME PIC X(264).
** First 12 bytes of user identifier from partner
15 MQCD-REMOTEUSERIDENTIFIER PIC X(12).
** Password from partner
15 MQCD-REMOTEPASSWORD PIC X(12).
** Ver:2 **
** Channel message retry exit name
15 MQCD-MSGRETRYEXIT PIC X(20).
** Channel message retry exit user data
15 MQCD-MSGRETRYUSERDATA PIC X(32).
** Number of times MCA will try to put the message, after first
** attempt has failed
15 MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
** Minimum interval in milliseconds after which the open or put
** operation will be retried
15 MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
** Ver:3 **
** Time in seconds between heartbeat flows
15 MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
** Batch duration
15 MQCD-BATCHINTERVAL PIC S9(9) BINARY.
** Speed at which nonpersistent messages are sent
15 MQCD-NONPERSISTENTMSGSPD PIC S9(9) BINARY.
** Length of MQCD structure
15 MQCD-STRUCLength PIC S9(9) BINARY.
** Length of exit name
15 MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
** Length of exit user data
15 MQCD-EXITDATALENGTH PIC S9(9) BINARY.
** Number of message exits defined
15 MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
** Number of send exits defined
15 MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
** Number of receive exits defined
15 MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
** Address of first MsgExit field
15 MQCD-MSGEXITPTR POINTER.
** Address of first MsgUserData field
15 MQCD-MSGUSERDATAPTR POINTER.
** Address of first SendExit field
15 MQCD-SENDEXITPTR POINTER.
** Address of first SendUserData field
15 MQCD-SENDUSERDATAPTR POINTER.
** Address of first ReceiveExit field
15 MQCD-RECEIVEEXITPTR POINTER.
** Address of first ReceiveUserData field
15 MQCD-RECEIVEUSERDATAPTR POINTER.
** Ver:4 **
** Address of a list of cluster names

```

```

15 MQCD-CLUSTERPTR POINTER.
** Number of clusters to which the channel belongs
15 MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
** Network priority
15 MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
** Ver:5 **
** Length of long MCA user identifier
15 MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
** Length of long remote user identifier
15 MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
** Address of long MCA user identifier
15 MQCD-LONGMCAUSERIDPTR POINTER.
** Address of long remote user identifier
15 MQCD-LONGREMOTEUSERIDPTR POINTER.
** MCA security identifier
15 MQCD-MCASECURITYID PIC X(40).
** Remote security identifier
15 MQCD-REMOTESECURITYID PIC X(40).
** Ver:6 **
** SSL CipherSpec
15 MQCD-SSLCIPHERSPEC PIC X(32).
** Address of SSL peer name
15 MQCD-SSLPEERNAMEPTR POINTER.
** Length of SSL peer name
15 MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
** Whether SSL client authentication is required
15 MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
** Keepalive interval
15 MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
** Local communications address
15 MQCD-LOCALADDRESS PIC X(48).
** Batch heartbeat interval
15 MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
** Ver:7 **
** Header data compression list
15 MQCD-HDRCOMPLIST PIC S9(9) BINARY.
** Message data compression list
15 MQCD-MSGCOMPLIST PIC S9(9) BINARY.
** Channel rank
15 MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
** Channel priority
15 MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
** Channel weight
15 MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
** Channel monitoring
15 MQCD-CHANNELMONITORING PIC S9(9) BINARY.
** Channel statistics
15 MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
** Ver:8 **
** Limit on sharing conversations
15 MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
** Message property control
15 MQCD-PROPERTYCONTROL PIC S9(9) BINARY.
** Limit on SVRCONN channel instances
15 MQCD-MAXINSTANCES PIC S9(9) BINARY.
** Limit on SVRCONN channel instances per client
15 MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
** Client channel weight
15 MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
** Connection affinity
15 MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
** Ver:9 **
** Batch data limit
15 MQCD-BATCHDATA LIMIT 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 **

```

Deklaracja RPG (ILE)

Ta deklaracja jest deklaracją RPG dla struktury MQCD.

```

D* MQCD Structure
D*
D* Channel definition name
D CDCHN          1      20
D* Structure version number

```

D	CDVER	21	24I 0
D*	Channel type		
D	CDCHT	25	28I 0
D*	Transport type		
D	CDTRT	29	32I 0
D*	Channel description		
D	CDDES	33	96
D*	Queue-manager name		
D	CDQM	97	144
D*	Transmission queue name		
D	CDXQ	145	192
D*	First 20 bytes of connection name		
D	CDSCN	193	212
D*	Reserved		
D	CDMCA	213	232
D*	LU 6.2 Mode name		
D	CDMOD	233	240
D*	LU 6.2 transaction program name		
D	CDTP	241	304
D*	Batch size		
D	CDBS	305	308I 0
D*	Disconnect interval		
D	CDDI	309	312I 0
D*	Short retry count		
D	CDSRC	313	316I 0
D*	Short retry wait interval		
D	CDSRI	317	320I 0
D*	Long retry count		
D	CDLRC	321	324I 0
D*	Long retry wait interval		
D	CDLRI	325	328I 0
D*	Channel security exit name		
D	CDSCX	329	348
D*	Channel message exit name		
D	CDMSX	349	368
D*	Channel send exit name		
D	CDSNX	369	388
D*	Channel receive exit name		
D	CDRCX	389	408
D*	Highest allowable message sequence number		
D	CDSNW	409	412I 0
D*	Maximum message length		
D	CDMML	413	416I 0
D*	Put authority		
D	CDPA	417	420I 0
D*	Data conversion		
D	CDDC	421	424I 0
D*	Channel security exit user data		
D	CDSCD	425	456
D*	Channel message exit user data		
D	CDMSD	457	488
D*	Channel send exit user data		
D	CDSND	489	520
D*	Channel receive exit user data		
D	CDRCD	521	552
D*	Ver:1 **		
D*	User identifier		
D	CDUID	553	564
D*	Password		
D	CDPW	565	576
D*	First 12 bytes of MCA user identifier		
D	CDAUI	577	588
D*	Message channel agent type		
D	CDCAT	589	592I 0
D*	Connection name		
D	CDCON	593	848
D	CDCN2	849	856
D*	First 12 bytes of user identifier from partner		
D	CDRUI	857	868
D*	Password from partner		
D	CDRPW	869	880
D*	Ver:2 **		
D*	Channel message retry exit name		
D	CDMRX	881	900
D*	Channel message retry exit user data		
D	CDMRD	901	932
D*	Number of times MCA will try to put the message, after first attempt has failed		
D	CDMRC	933	936I 0
D*	Minimum interval in milliseconds after which the open or put 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 **			

Deklaracja asemblera System/390

Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCD.

MQCD	DSECT	
MQCD_CHANNELNAME	DS	CL20 Channel definition name
MQCD_VERSION	DS	F Structure version number
MQCD_CHANNELTYPE	DS	F Channel type
MQCD_TRANSPORTTYPE	DS	F Transport type
MQCD_DESC	DS	CL64 Channel description
MQCD_QMGRNAME	DS	CL48 Queue-manager name
MQCD_XMITQNAME	DS	CL48 Transmission queue name
MQCD_SHORTCONNECTIONNAME	DS	CL20 First 20 bytes of connection name
*		
MQCD_MCANAME	DS	CL20 Reserved
MQCD_MODENAME	DS	CL8 LU 6.2 Mode name
MQCD_TPNAME	DS	CL64 LU 6.2 transaction program name
MQCD_BATCHSIZE	DS	F Batch size
MQCD_DISCINTERVAL	DS	F Disconnect interval
MQCD_SHORTRETRYCOUNT	DS	F Short retry count
MQCD_SHORTRETRYINTERVAL	DS	F Short retry wait interval
MQCD_LONGRETRYCOUNT	DS	F Long retry count
MQCD_LONGRETRYINTERVAL	DS	F Long retry wait interval
MQCD_SECURITYEXIT	DS	CLn Channel security exit name
MQCD_MSGEXIT	DS	CLn Channel message exit name
MQCD_SENDEXIT	DS	CLn Channel send exit name
MQCD_RECEIVEEXIT	DS	CLn Channel receive exit name
MQCD_SEQNUMBERWRAP	DS	F Highest allowable message sequence number
*		
MQCD_MAXMSGLENGTH	DS	F Maximum message length
MQCD_PUTAUTHORITY	DS	F Put authority
MQCD_DATACONVERSION	DS	F Data conversion
MQCD_SECURITYUSERDATA	DS	CL32 Channel security exit user data
MQCD_MSGUSERDATA	DS	CL32 Channel message exit user data
MQCD_SENDUSERDATA	DS	CL32 Channel send exit user data
MQCD_RECEIVEUSERDATA	DS	CL32 Channel receive exit user data
MQCD_USERIDENTIFIER	DS	CL12 User identifier
MQCD_PASSWORD	DS	CL12 Password

MQCD_MCAUSERIDENTIFIER	DS	CL12	First 12 bytes of MCA user identifier
* MQCD_MCATYPE	DS	F	Message channel agent type
MQCD_CONNECTIONNAME	DS	CL264	Connection name
MQCD_REMOTEUSERIDENTIFIER	DS	CL12	First 12 bytes of user identifier from partner
* MQCD_REMOTEPASSWORD	DS	CL12	Password from partner
MQCD_MSGRETRYEXIT	DS	CLn	Channel message retry exit name
MQCD_MSGRETRYUSERDATA	DS	CL32	Channel message retry exit user data
* MQCD_MSGRETRYCOUNT	DS	F	Number of times MCA will try to put the message, after the first attempt has failed
* MQCD_MSGRETRYINTERVAL	DS	F	Minimum interval in milliseconds after which the open or put operation will be retried
* MQCD_HEARTBEATINTERVAL	DS	F	Time in seconds between heartbeat flows
* MQCD_BATCHINTERVAL	DS	F	Batch duration
MQCD_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 Affinty
MQCD_BATCHDATA LIMIT	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)

Wizualna deklaracja podstawowa

Ta deklaracja jest deklaracją Visual Basic struktury MQCD.

W języku Visual Basic struktura MQCD może być używana razem ze strukturą MQCNO w wywołaniu MQCONN.

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		

Zmiana pól MQCD w wyjściu kanału

Wyjście kanału może zmienić pola na zmaterializowanych tabelach MQCD. Zmiany te nie są jednak zazwyczaj wykonywane, z wyjątkiem sytuacji wymienionych.

Jeśli program obsługi wyjścia kanału zmienia pole w strukturze danych MQCD, nowa wartość jest zwykle ignorowana przez proces kanału IBM MQ. Nowa wartość pozostaje jednak na zmaterializowanych tabelach MQCD i jest przekazywana do pozostałych wyjść w łańcuchu wyjścia i do dowolnej konwersacji współużytkującej instancję kanału.

Jeśli parametr SharingConversations ma wartość FALSE w strukturze MQCXP, zmiany w niektórych polach mogą być wykonywane w zależności od typu programu obsługi wyjścia, typu kanału oraz kodu przyczyny wyjścia. W poniższej tabeli przedstawiono pola, które można zmieniać i wpływają na zachowanie kanału oraz w jakich okolicznościach. Jeśli program obsługi wyjścia zmieni jedno z tych pól w innych okolicznościach lub dowolne pole, które nie zostało wyświetlone, nowa wartość zostanie zignorowana przez proces kanału. Nowa wartość pozostaje na zmaterializowanych tabelach MQCD i jest przekazywana do pozostałych wyjść w łańcuchu wyjścia i do dowolnej konwersacji współużytkującej instancję kanału.

Każdy typ programu obsługi wyjścia podczas wywołania inicjowania (MQXR_INIT) może zmienić wartość pola ChannelName dowolnego typu kanału, o ile parametr MQCXP SharingConversations ma wartość FALSE. Tylko wyjście zabezpieczeń może zmienić wartość w polu MCAUserIdentifier, niezależnie od wartości parametru MQCXP SharingConversations.

Pole	Kod przyczyny wyjścia	Typ wyjścia	Typ kanału
ChannelName	MQXR_INIT	Wszystkie	Wszystkie
TransportType	MQXR_INIT	Wszystkie	Wszystkie
XmitQName	MQXR_INIT	Wszystkie	SDR, RCVR
ModeName	MQXR_INIT	Wszystkie	Wszystkie
TpName	MQXR_INIT	Wszystkie	Wszystkie

Pole	Kod przyczyny wyjścia	Typ wyjścia	Typ kanału
BatchSize	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Liczba ShortRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Przedział czasu ShortRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Liczba LongRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Przedział czasu LongRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Zawijanie SeqNumber	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgDługość	MQXR_INIT	Wszystkie	Wszystkie
PutAuthority	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Wszystkie	Wszystkie
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Zabezpieczenia	RCVR, RQSTR, SVRCONN, CLUSRCVR

Pole	Kod przyczyny wyjścia	Typ wyjścia	Typ kanału
ConnectionName	MQXR_INIT	Wszystkie	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
Liczba MsgRetry	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
Przedział czasu MsgRetry	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Wszystkie	Wszystkie
BatchInterval	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Zabezpieczenia	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Wszystkie	Wszystkie
SSLPeerName	MQXR_INIT	Wszystkie	Wszystkie
Długość parametru SSLPeerName	MQXR_INIT	Wszystkie	Wszystkie
SSLClientAuth	MQXR_INIT	Wszystkie	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
Przedział czasu KeepAlive	MQXR_INIT	Wszystkie	Wszystkie
LocalAddress	MQXR_INIT	Wszystkie	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR
Lista HdrComp	MQXR_INIT	Wszystkie	Wszystkie

Pole	Kod przyczyny wyjścia	Typ wyjścia	Typ kanału
Lista MsgComp	MQXR_INIT	Wszystkie	Wszystkie
ChannelMonitoring	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Wszystkie	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP-parametr wyjścia kanału

Struktura MQCXP jest przekazywana do każdego typu wyjścia wywołanego przez agenta kanału komunikatów (Message Channel Agent-MCA), kanał połączenia klienckiego lub kanał połączenia z serwerem.

Patrz MQ_CHANNEL_EXIT.

Pola opisane jako "wejście do wyjścia" w opisach, które są zgodne, są ignorowane przez kanał, gdy wyjście zwraca element sterujący do kanału. Wszystkie pola wejściowe, które zmiany wyjścia w bloku parametrów wyjścia kanału nie zostaną zachowane podczas następnego wywołania. Zmiany wprowadzone w polach wejściowych/wyjściowych (na przykład w polu *ExitUserArea*) są zachowywane tylko w przypadku wywołań tej instancji tylko wyjścia. Takich zmian nie można używać do przekazywania danych między różnymi wyjściami zdefiniowanymi w tym samym kanale lub między tymi samymi wyjściami zdefiniowanymi w różnych kanałach.

Odsyłacze pokrewne

["Pola" na stronie 1610](#)

Ten temat zawiera listę wszystkich pól w strukturze MQCXP i opisuje poszczególne pola.

["Deklaracja C" na stronie 1622](#)

Ta deklaracja jest deklaracją C dla struktury MQCXP.

["Deklaracja języka COBOL" na stronie 1623](#)

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCXP.

["Deklaracja RPG \(ILE\)" na stronie 1623](#)

Ta deklaracja jest deklaracją RPG dla struktury MQCXP.

["Deklaracja assemblera System/390" na stronie 1624](#)

Ta deklaracja jest deklaracją assemblera System/390 dla struktury MQCXP.

Pola

Ten temat zawiera listę wszystkich pól w strukturze MQCXP i opisuje poszczególne pola.

StrucId (MQCHAR4)

To pole określa identyfikator struktury.

Wartość musi być następująca:

MQ_CXP_STRUC_ID

Identyfikator struktury parametru wyjścia kanału.

Dla języka programowania C jest również zdefiniowana stała zmienna MQ_CXP_STRUC_ID_ARRAY; ta stała ma taką samą wartość jak MQ_CXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe do wyjścia.

Wersja (MQLONG)

To pole określa numer wersji struktury.

Wartość zależy od środowiska:

MQ_CXP_VERSION_1

Struktura parametru wyjścia kanału Version-1 .

MQ_CXP_VERSION_2

Struktura parametru wyjścia kanału Version-2 .

Pole ma tę wartość w środowisku HP Integrity NonStop Server .

MQ_CXP_VERSION_3

Struktura parametru wyjścia kanału Version-3 .

Pole ma tę wartość w następujących środowiskach: Systemy UNIX nie znajduje się na liście w innym miejscu.

MQ_CXP_VERSION_4

Struktura parametru wyjścia kanału Version-4 .

MQ_CXP_VERSION_5

Struktura parametru wyjścia kanału Version-5 .

MQ_CXP_VERSION_6

Struktura parametru wyjścia kanału Version-6 .

MQ_CXP_VERSION_8

Struktura parametru wyjścia kanału Version-8 .

Pole ma tę wartość w następujących środowiskach: z/OS, AIX, HP-UX, Linux, IBM i, Solaris, Windows.

Pola, które istnieją tylko w najnowszych wersjach struktury, są identyfikowane jako takie w opisach pól.

Następująca stała określa numer wersji bieżącej wersji:

MQ_CXP_CURRENT_VERSION

Bieżąca wersja struktury parametru wyjścia kanału.

Wartość zależy od środowiska.

Uwaga: Gdy zostanie wprowadzona nowa wersja struktury MQ_CXP, układ istniejącej części nie jest zmieniany. Wyjście musi zatem sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji, która zawiera pola, które musi być używane przez wyjście.

To jest pole wejściowe do wyjścia.

ExitId (MQLONG)

To pole określa typ wywołanego wyjścia i jest ustawiony przy wpisach do procedury wyjścia.

Dozwolone są następujące wartości:

MQXT_CHANNEL_SEC_EXIT

Wyjście zabezpieczeń kanału.

MQXT_CHANNEL_MSG_EXIT

Wyjście komunikatu kanału.

MQXT_CHANNEL_SEND_EXIT

Wyjście wysyłania kanału.

MQXT_CHANNEL_RCV_EXIT

Wyjście odbierania kanału.

MQXT_CHANNEL_MSG_RETRY_EXIT

Wyjście komunikatu kanału-wyjście ponowienia.

MQXT_CHANNEL_AUTO_DEF_EXIT

Wyjście automatycznej definicji kanału.

W systemie z/OS ten typ wyjścia jest obsługiwany tylko dla kanałów typu MQCHT_CLUSSDR i MQCHT_CLUSRCVR.

To jest pole wejściowe do wyjścia.

ExitReason (MQLONG)

To pole określa przyczynę, dla której program obsługi wyjścia jest wywoływany i jest ustawiony przy wpisach do procedury wyjścia.

Nie jest on używany przez wyjście automatycznej definicji. Dozwolone są następujące wartości:

MQXR_INIT

Inicjowanie wyjścia.

Ta wartość wskazuje, że wyjście jest wywoływane po raz pierwszy. Pozwala ona wyjść na pozyskiwanie i inicjowanie dowolnych zasobów, których potrzebuje (na przykład: pamięci).

MQXR_TERM

Zakończ zakończenie.

Ta wartość wskazuje, że wyjście ma zostać zakończone. Program obsługi wyjścia powinien zwolnić wszystkie zasoby, które zostały przez niego pozyskane od momentu jego zainicjowania (na przykład: pamięć).

MQXR_MSG

Przetwórz komunikat.

Ta wartość wskazuje, że wyjście jest wywoływane w celu przetworzenia komunikatu. Ta wartość występuje tylko w przypadku wyjść komunikatów kanału.

MQXR_XMIT

Przetwórz transmisję.

Ta wartość występuje tylko w przypadku wyjścia wysyłania i odbierania kanału.

MQXR_SEC_MSG

Odebrano komunikat bezpieczeństwa.

Ta wartość występuje tylko w przypadku wyjść bezpieczeństwa kanału.

MQXR_INIT_SEC

Inicjowanie wymiany zabezpieczeń.

Ta wartość występuje tylko w przypadku wyjść bezpieczeństwa kanału.

Wyjście zabezpieczeń odbiornika jest zawsze wywoływane z tą przyczyną bezpośrednio po wywołaniu z MQXR_INIT, aby dać mu możliwość zainicjowania wymiany zabezpieczeń. Jeśli zostanie odtajona potencjalna transakcja (zwracając wartość MQXCC_OK zamiast MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG), wyjście zabezpieczeń nadawcy zostanie wywołane za pomocą MQXR_INIT_SEC.

Jeśli wyjście zabezpieczeń odbiornika zainicjuje wymianę zabezpieczeń (zwracając komendę MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG), wyjście zabezpieczeń nadawcy nie jest nigdy wywoływane za pomocą MQXR_INIT_SEC; zamiast tego jest wywoływane za pomocą MQXR_SEC_MSG w celu przetworzenia komunikatu odbiorcy. (W obu przypadkach jest to pierwsze wywołanie z MQXR_INIT.)

Jeśli jedno z wyjść zabezpieczeń nie zakończy żądań zakończenia kanału (przez ustawienie *ExitResponse* na MQXCC_SUPPRESS_FUNCTION lub MQXCC_CLOSE_CHANNEL), wymiana

zabezpieczeń musi zostać zakończona z boku, który zainicjował wymianę. Z tego powodu, jeśli wyjście zabezpieczeń jest wywoływane za pomocą komendy MQXR_INIT_SEC i inicjuje on wymianę, przy następnym wywołaniu wyjścia będzie on z opcją MQXR_SEC_MSG. Dzieje się tak, czy istnieje komunikat bezpieczeństwa dla wyjścia do przetworzenia, czy też nie. Jeśli partner zwraca wartość MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG, jest to komunikat zabezpieczeń, ale nie, jeśli partner zwraca wartość MQXCC_OK lub jeśli nie ma wyjścia zabezpieczeń dla partnera. Jeśli do przetworzenia nie ma komunikatu zabezpieczeń, wyjście zabezpieczeń na końcu inicjującym jest ponownie wywoływane z wartością *DataLength* równą zero.

MQXR_RETRY

Ponów próbę.

Ta wartość występuje tylko dla programów zewnętrznych ponowień komunikatu.

MQXR_AUTO_CLUSSDR

Automatyczna definicja kanału nadawczego klastra.

Ta wartość występuje tylko dla automatycznych wyjść definicji kanału.

MQXR_AUTO_RECEIVER

Automatyczna definicja kanału odbiorczego.

Ta wartość występuje tylko dla automatycznych wyjść definicji kanału.

MQXR_AUTO_SVRCONN

Automatyczna definicja kanału połączenia z serwerem.

Ta wartość występuje tylko dla automatycznych wyjść definicji kanału.

MQXR_AUTO_CLUSRCVR

Automatyczna definicja kanału odbiorczego klastra.

Ta wartość występuje tylko dla automatycznych wyjść definicji kanału.

MQXR_SEC_PARMS

Parametry bezpieczeństwa

Ta wartość ma zastosowanie tylko do wyjść zabezpieczeń i wskazuje, że struktura MQCSP jest przekazywana do wyjścia. Więcej informacji na ten temat zawiera sekcja [“MQCSP-parametry zabezpieczeń”](#) na stronie 336.

Uwaga:

1. Jeśli dla kanału zdefiniowano więcej niż jedno wyjście, są one wywoływane za pomocą MQXR_INIT, gdy inicjowane jest działanie agenta MCA. Ponadto są one wywoływane za pomocą MQXR_TERM, gdy agent MCA zostanie zakończony.
2. W przypadku wyjścia z automatycznego definiowania kanału program *ExitReason* nie jest ustawiony, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_4. W tym przypadku implikowana jest wartość MQXR_AUTO_SVRCONN.

To jest pole wejściowe do wyjścia.

ExitResponse (MQLONG)

To pole określa odpowiedź od wyjścia.

To pole jest ustawiane przez wyjście w celu komunikowania się z agentem MCA. Musi to być jedna z następujących wartości:

MQXCC_OK

Wyjście zostało zakończone pomyślnie.

- W przypadku wyjścia zabezpieczeń kanału wartość ta wskazuje, że przesyłanie komunikatów może być kontynuowane normalnie.
- W przypadku wyjścia dla ponowienia komunikatu kanału wartość ta wskazuje, że agent MCA musi czekać na przedział czasu zwrócony przez wyjście w polu *MsgRetryInterval* w tabeli MQCXP, a następnie ponowić próbę.

Pole *ExitResponse2* może zawierać dodatkowe informacje.

MQXCC_SUPPRESS_FUNCTION

Funkcja pomijania.

- W przypadku wyjścia zabezpieczeń kanału ta wartość wskazuje, że kanał musi zostać zakończony.
- W przypadku wyjścia komunikatów kanału wartość ta wskazuje, że komunikat nie ma być kontynuowany w kierunku jego miejsca docelowego. Zamiast tego agent MCA generuje komunikat raportu o wyjątku (jeśli został zażądany przez nadawcę oryginalnego komunikatu) i umieszcza komunikat znajdujący się w pierwotnym buforze w kolejce niedostarczonych komunikatów (jeśli nadawca określił wartość MQRO_DEAD_LETTER_Q) lub usuwa go (jeśli nadawca określił MQRO_DISCARD_MSG).

W przypadku komunikatów trwałych, jeśli nadawca określił wartość MQRO_DEAD_LETTER_Q, ale próba umieszczenia w kolejce niedostarczonych komunikatów nie powiedzie się lub nie ma kolejki niedostarczonych komunikatów, oryginalny komunikat jest pozostawiony w kolejce transmisji, a komunikat raportu nie jest generowany. Jeśli komunikat raportu nie może zostać pomyślnie wygenerowany, oryginalny komunikat jest również pozostawiany w kolejce transmisji.

Pole *Feedback* w strukturze MQDLH na początku komunikatu w kolejce niedostarczonych komunikatów wskazuje, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów. Ten kod sprzężenia zwrotnego jest również używany w deskrypcorze komunikatu dla komunikatu raportu o wyjątku (jeśli został on zażądany przez nadawcę).

- W przypadku wyjścia dla ponowienia komunikatu kanału wartość ta wskazuje, że agent MCA nie czeka i ponownie spróbuje ponowić komunikat. Zamiast tego agent MCA będzie kontynuował normalne przetwarzanie niepowodzenia (komunikat jest umieszczany w kolejce niedostarczonych komunikatów lub odrzucany, zgodnie z określonym przez nadawcę komunikatu).
- W przypadku wyjścia z automatycznego definiowania kanału należy określić wartość MQXCC_OK lub MQXCC_SUPPRESS_FUNCTION. Jeśli żadna z tych wartości nie zostanie określona, domyślnie przyjmowana jest wartość MQXCC_SUPPRESS_FUNCTION, a automatyczna definicja jest porzucona.

Ta odpowiedź nie jest obsługiwana dla kanałów wysyłania i odbierania kanału.

MQXCC_SEND_SEC_MSG

Wyślij komunikat bezpieczeństwa.

Tę wartość można ustawić tylko przez wyjście zabezpieczeń kanału. Wskazuje on, że wyjście udostępniło komunikat o zabezpieczeniu, który musi zostać przestany do partnera.

MQXCC_SEND_AND_REQUEST_SEC_MSG

Wyślij komunikat bezpieczeństwa, który wymaga odpowiedzi.

Tę wartość można ustawić tylko przez wyjście zabezpieczeń kanału. Wskazuje

- że wyjście udostępniło komunikat o zabezpieczeniu, który może zostać przekazany partnerowi, oraz
- że wyjście wymaga odpowiedzi od partnera. Jeśli nie zostanie odebrana żadna odpowiedź, kanał musi zostać zakończony, ponieważ program obsługi wyjścia nie podjął jeszcze decyzji, czy komunikacja może być kontynuowana.

MQXCC_SUPPRESS_EXIT

Pomijaj wyjście.

- Ta wartość może być ustawiona przez wszystkie typy wyjścia kanału inne niż wyjście zabezpieczeń lub wyjście automatyczne z definicją. Wyłącza on dalsze wywoływanie tego wyjścia (tak jakby jego nazwa była pusta w definicji kanału), aż do zakończenia kanału, gdy wyjście zostanie ponownie wywołane z *ExitReason* programu MQXR_TERM.
- Jeśli wyjście ponowienia komunikatu zwraca tę wartość, ponowne próby komunikatów dla kolejnych komunikatów są kontrolowane przez atrybuty kanału *MsgRetryCount* i *MsgRetryInterval* jako normalne. Dla bieżącego komunikatu agent MCA wykonuje liczbę oczekujących ponowień, w odstępach czasu podanych przez atrybut kanału *MsgRetryInterval*, ale tylko wtedy, gdy kod przyczyny jest taki, że agent MCA normalnie ponawiał próbę (patrz pole *MsgRetryCount* opisane w sekcji [“MQCD-definicja kanału” na stronie 1569](#)). Liczba oczekujących ponowień to wartość

atrybutu *MsgRetryCount* , pomniejszona o liczbę przypadków, w których wyjście zwróciło wartość MQXCC_OK dla bieżącego komunikatu. Jeśli ta liczba jest ujemna, MCA nie wykonuje żadnych dalszych prób dla bieżącego komunikatu.

MQXCC_CLOSE_CHANNEL

Zamknij kanał.

Tę wartość można ustawić za pomocą dowolnego typu wyjścia kanału z wyjątkiem wyjścia automatycznego definiowania.

Jeśli współużytkowanie konwersacji nie jest włączone, ta wartość zamyka kanał.

Jeśli współużytkowanie konwersacji jest włączone, ta wartość kończy konwersację. Jeśli ta rozmowa jest jedyną rozmową na kanale, kanał również się zamyka.

To pole jest polem wejścia/wyjścia z wyjścia.

ExitResponse2 (MQLONG)

To pole określa wtórną odpowiedź od wyjścia.

To pole jest ustawione na zero przy wpisie do procedury wyjścia. Program ten może zostać ustawiony przez wyjście w celu udostępnienia dodatkowych informacji dla funkcji kanału produktu IBM MQ . Nie jest on używany przez wyjście automatyczne definicji.

Wyjście może ustawić jedną lub więcej z poniższych wartości. Jeśli wymagane jest więcej niż jedno, wartości są dodawane. Podane kombinacje nie są poprawne; dozwolone są inne kombinacje.

MQXR2_PUT_WITH_DEF_ACTION

Umieść z działaniem domyślnym.

Ta wartość jest ustawiana przez wyjście komunikatów kanału odbiornika. Wskazuje on, że komunikat ma zostać umieszczony za pomocą domyślnego działania agenta MCA, który jest domyślnym identyfikatorem użytkownika agenta MCA, lub kontekstem *UserIdentifier* w deskrytorze MQMD (deskrytor komunikatu) komunikatu.

Wartość jest równa zero, co odpowiada wartości początkowej ustawionej po wywołaniu wyjścia. Stała jest udostępniana na potrzeby dokumentacji.

MQXR2_PUT_WITH_DEF_USERID

Umieść za pomocą domyślnego identyfikatora użytkownika.

Ta wartość może być ustawiona tylko przez wyjście komunikatu kanału odbiornika. Wskazuje on, że komunikat ma zostać umieszczony za pomocą domyślnego identyfikatora użytkownika agenta MCA.

MQXR2_PUT_WITH_MSG_USERID

Umieść za pomocą identyfikatora użytkownika komunikatu.

Ta wartość może być ustawiona tylko przez wyjście komunikatu kanału odbiornika. Wskazuje on, że komunikat ma zostać umieszczony z kontekstem *UserIdentifier* w deskrytorze MQMD (deskrytor komunikatu) komunikatu (może to być zmodyfikowane przez wyjście).

Należy ustawić tylko jedną z następujących wartości: MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERID i MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER

Użyj buforu agenta.

Ta wartość wskazuje, że wszystkie dane, które mają być przekazywane, znajdują się w *AgentBuffer* , a nie w *ExitBufferAddr* .

Wartość jest równa zero, co odpowiada wartości początkowej ustawionej po wywołaniu wyjścia. Stała jest udostępniana na potrzeby dokumentacji.

MQXR2_USE_EXIT_BUFFER

Użyj buforu wyjścia.

Ta wartość wskazuje, że wszystkie dane, które mają być przekazywane, znajdują się w *ExitBufferAddr* , a nie w *AgentBuffer* .

Należy ustawić tylko jeden z następujących wartości: MQXR2_USE_AGENT_BUFFER i MQXR2_USE_EXIT_BUFFER .

MQXR2_DEFAULT_CONTINUATION

Domyślna kontynuacja.

Kontynuacja z następnym wyjściem w łańcuchu zależy od odpowiedzi od wywołanego ostatniego wyjścia:

- Jeśli zostaną zwrócone wywołania MQXCC_SUPPRESS_FUNCTION lub MQXCC_CLOSE_CHANNEL, nie są wywoływane żadne dalsze wyjścia w łańcuchu.
- W przeciwnym razie wywoływane jest następne wyjście w łańcuchu.

MQXR2_CONTINUE_CHAIN

Przejdź do następnego wyjścia.

MQXR2_SUPPRESS_CHAIN

Pomiń pozostałe wyjścia w łańcuchu.

Jest to pole wejściowe/wyjściowe do wyjścia.

Opinia (MQLONG)

To pole określa kod sprzężenia zwrotnego.

To pole jest ustawione na wartość MQFB_NONE przy wpisie do procedury wyjścia.

Jeśli wyjście komunikatu kanału ustawia pole *ExitResponse* na MQXCC_SUPPRESS_FUNCTION, pole *Feedback* określa kod sprzężenia zwrotnego, który identyfikuje, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów), a także jest używany do wysyłania raportu o wyjątkach, jeśli został on zażądany. W takim przypadku, jeśli pole *Feedback* ma wartość MQFB_NONE, używany jest następujący kod sprzężenia zwrotnego:

MQFB_STOPPED_BY_MSG_EXIT

Komunikat został zatrzymany przez wyjście komunikatów kanału.

Wartość zwracana w tym polu przez wyjścia bezpieczeństwa kanału, wysyłania, odbierania i ponowienia komunikatu nie jest używana przez agenta MCA.

Wartość zwracana w tym polu przez wyjścia definicji automatycznego nie jest używana, jeśli *ExitResponse* jest wartością MQXCC_OK, ale w przeciwnym razie jest używana dla parametru *AuxErrorDataInt1* w komunikacie zdarzenia.

Jest to pole wejściowe/wyjściowe z wyjścia.

MaxSegmentDługość (MQLONG)

To pole określa maksymalną długość (w bajtach), która może zostać wysłana w jednej transmisji.

Nie jest on używany przez wyjście automatyczne definicji. Jest ono interesujące dla wyjścia wysyłania kanału, ponieważ to wyjście musi zapewnić, że nie zwiększy on wielkości segmentu transmisji do wartości większej niż *MaxSegmentLength*. Długość obejmuje początkowe 8 bajtów, których wyjście nie może ulec zmianie. Wartość jest negocjowana między funkcjami kanału IBM MQ , gdy kanał jest inicjowany. Więcej informacji na temat długości segmentów zawiera sekcja [Pisanie programów obsługi wyjścia kanału](#) .

Wartość w tym polu nie ma znaczenia, jeśli parametr *ExitReason* ma wartość MQXR_INIT.

To jest pole wejściowe do wyjścia.

Obszar ExitUser(MQBYTE16)

To pole określa obszar użytkownika wyjścia-pole dostępne dla wyjścia, które ma być używane.

Jest on inicjowany do zera binarnego przed pierwszym wywołaniem wyjścia (z zestawem *ExitReason* ustawionym na wartość MQXR_INIT), a następnie wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane w wywołaniach wyjścia.

Zdefiniowana jest następująca wartość:

MQXUA_NONE

Brak informacji o użytkowniku.

Wartość jest binarna zero dla długości pola.

Dla języka programowania C zdefiniowana jest również stała zmienna MQXUA_NONE_ARRAY; ta stała ma taką samą wartość co MQXUA_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_EXIT_USER_AREA_LENGTH. Jest to pole wejściowe/wyjściowe do wyjścia.

ExitData (MQCHAR32)

To pole określa dane wyjścia.

To pole jest ustawiane przy wpisowaniu do procedury wyjścia w celu uzyskania informacji o tym, że funkcje kanału IBM MQ zostały przejęte z definicji kanału. Jeśli takie informacje nie są dostępne, to pole jest puste.

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

To jest pole wejściowe do wyjścia.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

Liczba MsgRetry(MQLONG)

To pole określa, ile razy komunikat został ponowiony.

Przy pierwszym wywołaniu wyjścia dla konkretnego komunikatu, pole to ma wartość zero (nie próbowano jeszcze żadnych prób). Przy każdym kolejnym wywołaniu wyjścia dla tego komunikatu wartość ta jest zwiększana o jeden przez agenta MCA.

To jest pole wejściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli parametr *ExitReason* ma wartość MQXR_INIT. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

Przedział czasu MsgRetry(MQLONG)

To pole określa minimalny odstęp czasu (w milisekundach), po którym operacja put zostanie ponowiona.

Przy pierwszym wywołaniu wyjścia dla konkretnego komunikatu, pole to zawiera wartość atrybutu kanału *MsgRetryInterval*. Wyjście może pozostawić wartość bez zmian lub zmodyfikować ją w celu określenia innego przedziału czasu w milisekundach. Jeśli wyjście zwraca wartość MQXCC_OK w programie *ExitResponse*, agent MCA czeka przez co najmniej ten przedział czasu przed ponowieniem operacji MQOPEN lub MQPUT. Podany przedział czasu musi być równy zero lub większy.

Po drugim i kolejnych uruchomieniu wyjścia dla tego komunikatu pole to zawiera wartość zwracaną przez poprzednie wywołanie wyjścia.

Jeśli wartość zwrócona w polu *MsgRetryInterval* jest mniejsza niż zero lub większa niż 999 999 999, a *ExitResponse* to MQXCC_OK, agent MCA ignoruje pole *MsgRetryInterval* w MQCXP i oczekuje zamiast przedziału czasu określonego przez atrybut kanału *MsgRetryInterval*.

Jest to pole wejściowe/wyjściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli parametr *ExitReason* ma wartość MQXR_INIT. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

MsgRetryPrzyczyna (MQLONG)

To pole określa kod przyczyny z poprzedniej próby umieszczenia komunikatu.

To pole jest kodem przyczyny z poprzedniej próby umieszczenia komunikatu. Jest to jedna z wartości MQRC_*.

To jest pole wejściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli parametr *ExitReason* ma wartość MQXR_INIT. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

HeaderLength (MQLONG)

To pole określa długość informacji nagłówka.

To pole jest istotne tylko dla wyjścia komunikatu i wyjścia dla ponowienia komunikatu. Wartość określa długość struktury nagłówka routingu na początku danych komunikatu. Są to: struktura MQXQH, MQMDE (nagłówek rozszerzenia opisu komunikatu) i (dla komunikatu z listą dystrybucyjną) struktura MQDH i tablice rekordów MQOR i MQPMR, które są zgodne ze strukturą MQXQH.

Wyjście komunikatu może sprawdzić te informacje nagłówka i w razie potrzeby zmodyfikować je, ale dane zwracane przez wyjście muszą być w poprawnym formacie. Wyjście nie może na przykład szyfrować lub kompresować danych nagłówka na końcu wysyłania, nawet jeśli wyjście komunikatu na końcu odbierającego powoduje kompensację zmian.

Jeśli wyjście komunikatu modyfikuje informacje nagłówka w taki sposób, aby zmienić jego długość (na przykład przez dodanie innego miejsca docelowego do komunikatu listy dystrybucyjnej), musi ona odpowiednio zmienić wartość *HeaderLength* przed zwróceniem.

Jest to pole wejściowe/wyjściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli parametr *ExitReason* ma wartość MQXR_INIT. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

PartnerName (MQCHAR48)

To pole określa nazwę partnera.

Imię i nazwisko współnika, jak następuje:

- W przypadku kanałów SVRCONN jest to identyfikator zalogowanego użytkownika na kliencie.
- Dla wszystkich innych typów kanału jest to nazwa menedżera kolejek partnera.

Po zainicjowaniu wyjścia to pole jest puste, ponieważ menedżer kolejek nie zna nazwy partnera, dopóki nie zostanie rozpoczęte początkowe negocjacje.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

FAPLevel (MQLONG)

Wynegocjowane formaty i poziom protokołów.

To jest pole wejściowe do wyjścia. Zmiany w tym polu powinny być wprowadzane tylko pod nadzorem usługi IBM. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

CapabilityFlags (MQLONG)

To pole określa opcje możliwości.

Zdefiniowane są następujące elementy:

MQCF_NONE

Brak flag.

MQCF_DIST_LISTS

Obsługiwane są listy dystrybucyjne.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

ExitNumber (MQLONG)

To pole określa numer porządkowy wyjścia.

Numer porządkowy wyjścia w ramach typu zdefiniowanego w *ExitId*. Na przykład, jeśli wywoływane wyjście jest trzecim zdefiniowanym wyjściem komunikatu, to pole zawiera wartość 3. Jeśli typ wyjścia to jeden, dla którego nie można zdefiniować listy wyjść (na przykład wyjście bezpieczeństwa), to pole ma wartość 1.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_5.

ExitSpace (MQLONG)

To pole określa liczbę bajtów w buforze transmisji zarezerwowanych dla wyjścia, które ma zostać użyte.

To pole ma znaczenie tylko w przypadku wyjścia wysyłania. Określa on ilość miejsca w bajtach, jaką pełni funkcję rezerwowania funkcji kanału IBM MQ w buforze transmisji dla wyjścia, które ma być używane. To pole umożliwia wyjście w celu dodania do buforu transmisji niewielkiej ilości danych (zwykle nie więcej niż kilkaset bajtów) do wykorzystania przez komplementarne wyjście odbierania na drugim końcu. Dane dodane przez wyjście wysyłania muszą zostać usunięte przez wyjście odbierania.

Wartość zawsze wynosi zero w z/OS.

Uwaga: Ta funkcja nie może być używana do wysyłania dużych ilości danych, ponieważ może to pogorszać wydajność, a nawet nie hamować działania kanału.

Ustawiając wartość *ExitSpace*, wyjście jest gwarantowane, że w buforze transmisji jest zawsze co najmniej taka liczba bajtów, aby wyjście było używane. Jednak wyjście może być mniejsze niż zarezerwowana kwota lub większa niż ilość zarezerwowana w przypadku miejsca dostępnego w buforze transmisji. Miejsce wyjścia w buforze jest udostępniane zgodnie z istniejącymi danymi.

Program *ExitSpace* może zostać ustawiony przez wyjście tylko wtedy, gdy parametr *ExitReason* ma wartość MQXR_INIT; we wszystkich innych przypadkach wartość zwrócona przez wyjście jest ignorowana. W przypadku wejścia do wyjścia parametr *ExitSpace* ma wartość zero dla wywołania MQXR_INIT i jest to wartość zwracana przez wywołanie MQXR_INIT w innych przypadkach.

Jeśli wartość zwrócona przez wywołanie MQXR_INIT jest ujemna lub jest mniej niż 1024 bajty dostępne w buforze transmisji dla danych komunikatu po ponownym obsługiwaniu żądanego obszaru wyjścia dla wszystkich wyjść nadawanych w łańcuchu, agent MCA wyświetli komunikat o błędzie i zamknie kanał. Podobnie, jeśli podczas przesyłania danych wyjście w łańcuchu wyjścia wysyłania przydzieli więcej przestrzeni użytkownika niż zarezerwowane, tak aby w buforze transmisji danych komunikatu pozostało mniej niż 1024 bajty, agent MCA wyświetli komunikat o błędzie i zamknie kanał. Limit 1024 pozwala na przetwarzanie przepływów sterowania i administracyjnych kanału przez łańcuch wyjść nadawanych, bez potrzeby segmentacji przepływów.

Jest to pole wejściowe/wyjściowe do wyjścia, jeśli parametr *ExitReason* ma wartość MQXR_INIT, a pole wejściowe we wszystkich innych przypadkach. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_5.

Identyfikator SSLCertUser(MQCHAR12)

W tym polu jest określony parametr UserId powiązany ze zdalnym certyfikatem.

Jest pusta na wszystkich platformach z wyjątkiem z/OS

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6.

SSLRemCertIssName, Długość (MQLONG)

To pole określa długość (w bajtach) pełnej nazwy wyróżniającej wystawcy certyfikatu zdalnego wskazanego przez parametr SSLCertRemoteIssuerNamePtr.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6. Wartość jest równa zero, jeśli nie jest to kanał SSL.

SSLRemCertIssNamePtr (PMQVOID)

W tym polu podaje się adres pełnej nazwy wyróżniającej wystawcy certyfikatu zdalnego.

Jego wartością jest pusty wskaźnik, jeśli nie jest to kanał SSL.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6.

Uwaga: Zachowanie zabezpieczeń kanału kończy się przy określaniu nazwy wyróżniającej podmiotu, a nazwa wyróżniająca wystawcy została zmieniona w wersji IBM MQ v7.1. Więcej informacji na ten temat zawiera sekcja [Programy obsługi wyjścia zabezpieczeń kanału](#).

SecurityParms (PMQCSP)

To pole służy do określania adresu struktury MQCSP używanej do określania identyfikatora użytkownika i hasła.

Wartością początkową tego pola jest wskaźnik pusty.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6.

Wartość w tym polu, która jest zwracana przez wyjście, musi być użyteczna przez produkt IBM MQ do momentu wywołania MQXR_TERM.

Kompresja CurHdr(MQLONG)

To pole określa, która technika jest obecnie używana do kompresowania danych nagłówka.

Jest ona ustawiona na jedną z następujących wartości:

MQCOMPRESS_NONE

Dane nagłówka nie są kompresowane.

MQCOMPRESS_SYSTEM

Dane nagłówka są kompresowane.

Wartość może zostać zmieniona przez wyjście komunikatu kanału wysyłającego do jednej z wynegocjowanych obsługiwanych wartości, do których dostęp jest uzyskiwany z pola HdrComp(Lista HdrComp) na dysku MQCD. Umożliwia to użycie techniki kompresji danych nagłówka, które mają być wybrane dla każdego komunikatu na podstawie treści komunikatu. Zmieniona wartość jest używana tylko dla bieżącego komunikatu. Kanał zostanie zakończony, jeśli atrybut zostanie zmieniony na nieobsługiwaną wartość. Wartość ta jest ignorowana, jeśli zostanie zmieniona poza wyjściem komunikatu kanału wysyłającego.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6.

Kompresja CurMsg(MQLONG)

To pole określa, która technika jest obecnie używana do kompresowania danych komunikatu.

Jest ona ustawiona na jedną z następujących wartości:

MQCOMPRESS_NONE

Dane nagłówka nie są kompresowane.

MQCOMPRESS_RLE

Kompresja danych komunikatu jest wykonywana przy użyciu kodowania grupowego.

MQCOMPRESS_ZLIBFAST

Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowana jest szybka kompresja.

MQCOMPRESS_ZLIBHIGH

Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowany jest wysoki poziom kompresji.

Wartość może zostać zmieniona przez wyjście komunikatu kanału wysyłającego do jednej z wynegocjowanych obsługiwanych wartości, do których dostęp jest uzyskiwany z pola listy MsgCompz tabeli MQCD. Umożliwia to użycie techniki kompresji danych komunikatu, które będą decydowały o każdym komunikacie na podstawie treści komunikatu. Zmieniona wartość jest używana tylko dla bieżącego komunikatu. Kanał zostanie zakończony, jeśli atrybut zostanie zmieniony na nieobsługiwaną wartość. Wartość ta jest ignorowana, jeśli zostanie zmieniona poza wyjściem komunikatu kanału wysyłającego.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_6.

Hconn (MQHCONN)

To pole określa uchwyt połączenia używany przez wyjście, jeśli wymaga on wykonania wszystkich wywołań MQI w ramach wyjścia.

To pole nie ma znaczenia dla wyjść działających na kanałach połączeń z klientem, gdzie zawiera wartość MQHC_UNUSABLE_HCONN (-1).

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_7.

SharingConversations (MQBOOL)

To pole określa, czy konwersacja jest jedyną, która może być obecnie uruchomiona w tej instancji kanału, czy też może być uruchomiona więcej niż jedna konwersacja w tej instancji kanału.

Wskazuje również, czy program obsługi wyjścia jest narażony na ryzyko zmiany MQCD przez inny program obsługi wyjścia działający w tym samym czasie.

To pole ma zastosowanie tylko w przypadku programów obsługi wyjścia działających w kanałach połączeń typu klient lub serwer.

Jest ona ustawiona na jedną z następujących wartości:

FAŁSZ

Instancja wyjścia jest jedyną instancją wyjściową, która może być obecnie uruchomiona w tej instancji kanału. Pozwala to na bezpieczne aktualizowanie pól MQCD bez rywalizacji z innymi wyjściami działającymi w innych instancjach kanału. To, czy zmiany w polach MQCD są wykonywane przez kanał, jest definiowane przez tabelę pól MQCD w produkcie [“Zmiana pól MQCD w wyjściu kanału” na stronie 1607](#).

PRAWDA

Instancja wyjścia nie jest jedyną instancją wyjścia, która może być obecnie uruchomiona w tej instancji kanału. Wszystkie zmiany wprowadzone w tabeli MQCD nie są wykonywane przez kanał, z wyjątkiem zmian wymienionych w tabeli w polach MQCD w produkcie [“Zmiana pól MQCD w wyjściu kanału” na stronie 1607](#) z przyczyn wyjścia innych niż MQXR_INIT. Jeśli to wyjście aktualizuje pola MQCD, upewnij się, że nie ma rywalizacji z innymi wyjściami uruchamianych w innych konwersacjach w tym samym czasie, udostępniając serializację między wyjściami, które są uruchamiane w tej instancji kanału.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_7.

MCAUserSource (MQLONG)

To pole określa źródło podanego identyfikatora użytkownika MCA.

Może zawierać jedną z następujących wartości:

MQUSRC_MAP

Identyfikator użytkownika jest określony w atrybucie MCAUSER.

MQUSRC_CHANNEL

Identyfikator użytkownika jest przepływowym od partnera przychodzącego lub określony w polu MCAUSER zdefiniowanym w obiekcie kanału.

To jest pole wejściowe do wyjścia. To pole nie jest obecne, jeśli wersja jest mniejsza niż MQCXP_VERSION_8.

Punkty pEntry(PMQIEP)

To pole określa adres punktu wejścia interfejsu dla wywołania MQI lub DCI.

Pole nie jest obecne, jeśli *Wersja* jest mniejsza niż MQCXP_VERSION_8.

RemoteProduct (MQCHAR4)

To pole określa nazwę produktu zdalnego.

To pole identyfikuje zdalny produkt klienta, na przykład C lub Java, który jest wyświetlany w polu **RPRODUCT** w polu WYŚWIETL CHSTATUS.

Pole nie jest obecne, jeśli *Wersja* jest mniejsza niż MQCXP_VERSION_9.

RemoteVersion (MQCHAR8)

To pole określa nazwę wersji zdalnej.

To pole określa wersję bibliotek klienta, która jest wyświetlana w polu **RVERSION** (DISPLAY CHSTATUS).

Pole nie jest obecne, jeśli *Wersja* jest mniejsza niż MQCXP_VERSION_9.

Deklaracja C

Ta deklaracja jest deklaracją C dla struktury MQCXP.

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     ExitId;           /* Type of exit */
    MQLONG     ExitReason;       /* Reason for invoking exit */
    MQLONG     ExitResponse;     /* Response from exit */
    MQLONG     ExitResponse2;    /* Secondary response from exit */
    MQLONG     Feedback;        /* Feedback code */
    MQLONG     MaxSegmentLength; /* Maximum segment length */
    MQBYTE16   ExitUserArea;     /* Exit user area */
    MQCHAR32   ExitData;        /* Exit data */
    MQLONG     MsgRetryCount;    /* Number of times the message has been
                                retrieved */
    MQLONG     MsgRetryInterval; /* Minimum interval in milliseconds after
                                which the put operation should be
                                retrieved */
    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 */
    MQCHAR4    RemoteProduct;    /* The identifier for the remote product */
    MQCHAR8    RemoteVersion;    /* The version of the remote product */
    /* Ver:9 */
};
```

Deklaracja języka COBOL

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCXP.

```
** MQCXP structure
10 MQCXP.
** Structure identifier
15 MQCXP-STRUCID PIC X(4).
** Structure version number
15 MQCXP-VERSION PIC S9(9) BINARY.
** Type of exit
15 MQCXP-EXITID PIC S9(9) BINARY.
** Reason for invoking exit
15 MQCXP-EXITREASON PIC S9(9) BINARY.
** Response from exit
15 MQCXP-EXITRESPONSE PIC S9(9) BINARY.
** Secondary response from exit
15 MQCXP-EXITRESPONSE2 PIC S9(9) BINARY.
** Feedback code
15 MQCXP-FEEDBACK PIC S9(9) BINARY.
** Maximum segment length
15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
** Exit user area
15 MQCXP-EXITUSERAREA PIC X(16).
** Exit data
15 MQCXP-EXITDATA PIC X(32).
** Number of times the message has been retrieved
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.
```

Deklaracja RPG (ILE)

Ta deklaracja jest deklaracją RPG dla struktury MQCXP.

```
D*..1.....2.....3.....4.....5.....6.....7..
D* MQCXP Structure
D*
D* Structure identifier
D CXSID 1 4
D* Structure version number
D CXVER 5 8I 0
D* Type of exit
```

D	CXXID	9	12I	0
D*	Reason for invoking exit			
D	CXREA	13	16I	0
D*	Response from exit			
D	CXRES	17	20I	0
D*	Secondary response from exit			
D	CXRE2	21	24I	0
D*	Feedback code			
D	CXFB	25	28I	0
D*	Maximum segment length			
D	CXMSL	29	32I	0
D*	Exit user area			
D	CXUA	33	48	
D*	Exit data			
D	CXDAT	49	80	
D*	Number of times the message has been retried			
D	CXMRC	81	84I	0
D*	Minimum interval in milliseconds after which the put operation 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

Deklaracja asemblera System/390

Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCXP.

MQCXP	DSECT		
MQCXP_STRUCID	DS	CL4	Structure identifier
MQCXP_VERSION	DS	F	Structure version number
MQCXP_EXITID	DS	F	Type of exit
MQCXP_EXITREASON	DS	F	Reason for invoking exit
MQCXP_EXITRESPONSE	DS	F	Response from exit
MQCXP_EXITRESPONSE2	DS	F	Secondary response from exit
MQCXP_FEEDBACK	DS	F	Feedback code
MQCXP_MAXSEGMENTLENGTH	DS	F	Maximum segment length
MQCXP_EXITUSERAREA	DS	XL16	Exit user area
MQCXP_EXITDATA	DS	CL32	Exit data
MQCXP_MSGRETRYCOUNT	DS	F	Number of times the message has been retried
*			
MQCXP_MSGRETRYINTERVAL	DS	F	Minimum interval in milliseconds after which the put operation should be retried
*			
MQCXP_MSGRETRYREASON	DS	F	Reason code from previous attempt to put the message
*			
MQCXP_HEADERLENGTH	DS	F	Length of header information
MQCXP_PARTNERNAME	DS	CL48	Partner Name
MQCXP_FAPLEVEL	DS	F	Negotiated Formats and Protocols level
*			

MQCXP_CAPABILITYFLAGS	DS	F	Capability flags
MQCXP_EXITNUMBER	DS	F	Exit number
MQCXP_EXITSPLACE	DS	F	Number of bytes in transmission buffer reserved for exit to use
* MQCXP_SSLCERTUSERID	DS	CL12	User identifier associated with remote SSL certificate
* MQCXP_SSLREMCERTISSNAMELENGTH	DS	F	Length of distinguished name of issuer of remote SSL certificate
* MQCXP_SSLREMCERTISSNAMEPTR	DS	F	Address of distinguished name of issuer of remote SSL certificate
* MQCXP_SECURITYPARMS	DS	F	Address of security parameters
MQCXP_CURHDRCOMPRESSION	DS	F	Header data compression used for current message
* MQCXP_CURMSGCOMPRESSION	DS	F	Message data compression used for current message
* MQCXP_HCONN	DS	F	Connection handle
MQCXP_SHARINGCONVERSATIONS	DS	F	Multiple conversations possible on channel inst?
* MQCXP_MCAUSERSOURCE	DS	F	Source of the provided MCA user ID
MQCXP_LENGTH	EQU	*-MQCXP	
	ORG	MQCXP	
MQCXP_AREA	DS	CL(MQCXP_LENGTH)	

MQXWD-deskryptor oczekiwania wyjścia

Struktura MQXWD jest parametrem wejściowym/wyjściowym w wywołaniu MQXWAIT.

Ta struktura jest obsługiwana tylko w systemie z/OS.

Odsyłacze pokrewne

[“Pola” na stronie 1625](#)

Ten temat zawiera listę wszystkich pól w strukturze MQXWD i opisuje poszczególne pola.

[“Deklaracja C” na stronie 1626](#)

Ta deklaracja jest deklaracją C dla struktury MQXWD.

[“Deklaracja asemblera System/390” na stronie 1626](#)

Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQXWD.

Pola

Ten temat zawiera listę wszystkich pól w strukturze MQXWD i opisuje poszczególne pola.

StrucId (MQCHAR4)

To pole określa identyfikator struktury.

Wartość musi być następująca:

MQXWD_STRUC_ID

Identyfikator struktury deskryptora oczekiwania wyjścia.

Dla języka programowania C zdefiniowana jest również stała MQXWD_STRUC_ID_ARRAY; ta stała ma taką samą wartość jak MQXWD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Początkowa wartość tego pola to MQXWD_STRUC_ID.

Wersja (MQLONG)

To pole określa numer wersji struktury.

Wartość musi być następująca:

MQXWD_VERSION_1

Numer wersji struktury deskryptora oczekiwania wyjścia.

Początkowa wartość tego pola to MQXWD_VERSION_1.

Reserved1 (MQLONG)

To pole jest zarezerwowane. Jego wartość musi wynosić zero.

To jest pole wejściowe.

Reserved2 (MQLONG)

To pole jest zarezerwowane. Jego wartość musi wynosić zero.

To jest pole wejściowe.

Reserved3 (MQLONG)

To pole jest zarezerwowane. Jego wartość musi wynosić zero.

To jest pole wejściowe.

EBC (MQLONG)

To pole określa blok sterujący zdarzenia, na który ma być czekać.

To pole jest blokiem sterowania zdarzeniami (ECB), na którym należy czekać. Przed wywołaniem wywołania MQXWAIT musi on mieć wartość zero; po pomyślnym zakończeniu zawiera kod pocztowy.

To pole jest polem wejścia/wyjścia.

Deklaracja C

Ta deklaracja jest deklaracją C dla struktury MQXWD.

```
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
    MQCHAR4  StrucId;      /* Structure identifier */
    MQLONG   Version;      /* Structure version number */
    MQLONG   Reserved1;    /* Reserved */
    MQLONG   Reserved2;    /* Reserved */
    MQLONG   Reserved3;    /* Reserved */
    MQLONG   ECB;          /* Event control block to wait on */
};
```

Deklaracja assemblera System/390

Ta deklaracja jest deklaracją assemblera System/390 dla struktury MQXWD.

```
MQXWD          DSECT
MQXWD_STRUCID  DS    CL4  Structure identifier
MQXWD_VERSION  DS    F    Structure version number
MQXWD_RESERVED1 DS    F    Reserved
MQXWD_RESERVED2 DS    F    Reserved
MQXWD_RESERVED3 DS    F    Reserved
MQXWD_ECB      DS    F    Event control block to wait on
*
MQXWD_LENGTH   EQU    *-MQXWD
               ORG    MQXWD
MQXWD_AREA     DS     CL(MQXWD_LENGTH)
```

Wywołania wyjścia obciążenia klastra i struktury danych

W tej sekcji znajdują się informacje uzupełniające dotyczące wyjścia obciążenia klastra i struktur danych. Jest to ogólne informacje na temat interfejsu programistycznego.

Istnieje możliwość zapisu wyjść obciążenia klastra w następujących językach programowania:

- C
- Assembler System/390 (IBM MQ for z/OS)

Wywołanie jest opisane w:

- [“MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia” na stronie 1627](#)

Typy danych struktury używane przez wyjście są opisane w:

- [“MQXCLWLN -Nawiguj rekordy obciążenia klastra” na stronie 1629](#)
- [“MQWXP -Struktura parametru wyjścia obciążenia klastra” na stronie 1632](#)
- [“MQWDR-Struktura rekordu miejsca docelowego obciążenia klastra” na stronie 1641](#)

- [“MQWQR -Struktura rekordu kolejki obciążenia klastra” na stronie 1645](#)
- [“MQWCR -Struktura rekordu klastra obciążenia klastra” na stronie 1650](#)
-  [Asynchroniczne działanie komend CLUSTER w systemie z/OS](#)

W tej sekcji atrybuty menedżera kolejek i atrybuty kolejek są wyświetlane w całości. Nazwy równoważne, które są używane w komendach MQSC, są przedstawione poniżej. Szczegółowe informacje na temat komend MQSC zawiera sekcja [Komendy MQSC](#).

Tabela 193. Atrybuty menedżera kolejek	
Imię i nazwisko	Nazwa używana w MQSC
<i>ClusterWorkloadData</i>	CLWLDATA
<i>ClusterWorkloadExit</i>	CLWLEXIT
<i>ClusterWorkloadLength</i>	CLWLLEN

Tabela 194. Kolejka - atrybuty	
Imię i nazwisko	Nazwa używana w MQSC
<i>DefBind</i>	DEFBIND
<i>DefPersistence</i>	DEFPSIST
<i>DefPriority</i>	DEFPRTY
<i>InhibitPut</i>	PUT
<i>QDesc</i>	DESCR

Informacje pokrewne

[Pisanie i kompilowanie wyjść obciążenia klastra](#)

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

Uwaga: Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_CLUSTER_WORKLOAD_EXIT . Zamiast tego nazwa wyjścia obciążenia klastra jest definiowana za pomocą atrybutu menedżera kolejek *ClusterWorkloadExit* .

Wyjście MQ_CLUSTER_WORKLOAD_EXIT jest obsługiwane na wszystkich platformach.

Składnia

MQ_CLUSTER_WORKLOAD_EXIT (*ExitParms*)

Odsyłacze pokrewne

[MQXCLWLN](#) -Nawiguj rekordy obciążenia klastra

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

[MQWXP](#) -Struktura parametru wyjścia obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

[MQWDR](#)-Struktura rekordu miejsca docelowego obciążenia klastra

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

Parametry dla MQ_CLUSTER_WORKLOAD_EXIT

Opis parametrów w wywołaniu MQ_CLUSTER_WORKLOAD_EXIT .

ExitParms (MQWXP) -input/output

Wyjdź z bloku parametrów.

- Wyjście ustawia informacje w programie MQWXP , aby wskazać sposób zarządzania obciążeniem.

Odsyłacze pokrewne

Użycie notatek

Funkcja wykonywana przez wyjście obciążenia klastra jest zdefiniowana przez dostawcę wyjścia. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w powiązonym bloku kontrolnym MQWXP.

Wywołania języka dla produktu MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Użycie notatek

Funkcja wykonywana przez wyjście obciążenia klastra jest zdefiniowana przez dostawcę wyjścia. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w powiązonym bloku kontrolnym MQWXP.

Menedżer kolejek nie udostępnił punktu wejścia o nazwie MQ_CLUSTER_WORKLOAD_EXIT . Jednak typedef jest dostarczany dla nazwy MQ_CLUSTER_WORKLOAD_EXIT w języku programowania C. Użyj typu typedef, aby zadeklarować wyjście napisane przez użytkownika, aby upewnić się, że parametry są poprawne.

Odsyłacze pokrewne

Parametry dla MQ_CLUSTER_WORKLOAD_EXIT

Opis parametrów w wywołaniu MQ_CLUSTER_WORKLOAD_EXIT .

Wywołania języka dla produktu MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Wywołania języka dla produktu MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Wywołanie C

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Zastąp parametr *MQ_CLUSTER_WORKLOAD_EXIT* nazwą funkcji wyjścia obciążenia klastra.

Zadeklaruj parametry *MQ_CLUSTER_WORKLOAD_EXIT* w następujący sposób:

```
MQWXP ExitParms; /* Exit parameter block */
```

Wywołanie High Level Assembler

```
CALL EXITNAME,(EXITPARMS)
```

Zadeklaruj parametry w następujący sposób:

```
EXITPARMS      CMQWXP      Exit parameter block
```


Odsyłacze pokrewne

[Parametry dla MQ_CLUSTER_WORKLOAD_EXIT](#)

Opis parametrów w wywołaniu MQ_CLUSTER_WORKLOAD_EXIT .

Użycie notatek

Funkcja wykonywana przez wyjście obciążenia klastra jest zdefiniowana przez dostawcę wyjścia. Wyjście musi jednak być zgodne z regułami zdefiniowanymi w powiązonym bloku kontrolnym MQWXP.

MQXCLWLN -Nawiguj rekordy obciążenia klastra

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

Pamięć podręczna klastra to obszar pamięci głównej używany do przechowywania informacji związanych z klastrem.

Jeśli pamięć podręczna klastra jest statyczna, ma stałą wielkość. Jeśli ustawisz ją na dynamiczną, pamięć podręczna klastra będzie mogła się rozwijać zgodnie z wymaganiami.

Ustaw typ pamięci podręcznej klastra na STATIC lub DYNAMIC , używając parametru systemowego lub makra.

- Parametr systemowy ClusterCacheType na platformach innych niż z/OS
- Parametr CLCACHE w makrze CSQ6SYSP w systemie z/OS.

Składnia

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Odsyłacze pokrewne

[MQ_CLUSTER_WORKLOAD_EXIT](#) -opis połączenia

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

[MQWXP](#) -Struktura parametru wyjścia obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

[MQWDR](#)-Struktura rekordu miejsca docelowego obciążenia klastra

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

[MQWQR](#) -Struktura rekordu kolejki obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

[MQWCR](#) -Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

Parametry dla MQXCLWLN -nawigowanie rekordów obciążenia klastra

Opis parametrów w wywołaniu MQXCLWLN .

ExitParms (MQWXP) -input/output

Wyjdź z bloku parametrów.

Struktura ta zawiera informacje związane z wywoływaniem wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać sposób zarządzania obciążeniem.

CurrentRecord (MQPTR) -wejście

Adres bieżącego rekordu.

Struktura ta zawiera informacje dotyczące adresu rekordu aktualnie sprawdzanego przez wyjście. Rekord musi być jednym z następujących typów:

- Rekord miejsca docelowego obciążenia klastra (MQWDR)
- Rekord kolejki obciążenia klastra (MQWQR)

- Rekord klastra obciążenia klastra (MQWCR)

***NextOffset* (MQLONG) -wejście**

Przesunięcie następnego rekordu.

Struktura ta zawiera informacje odnoszące się do przesunięcia następnego rekordu lub struktury. *NextOffset* jest wartością odpowiedniego pola przesunięcia w bieżącym rekordzie i musi być jednym z następujących pól:

- Pole ChannelDefPrzesunięcie w Komponent MQWDR
- Pole ClusterRecOffset w obszarze MQWDR
- Pole ClusterRecOffset w obszarze MQWQR
- Pole ClusterRecOffset w MQWCR

***NextRecord* (MQPTR) -wyjście**

Adres następnego rekordu lub struktury.

Struktura ta zawiera informacje odnoszące się do adresu następnego rekordu lub struktury. Jeśli *CurrentRecord* jest adresem MQWDR, a *NextOffset* jest wartością pola ChannelDefOffset , *NextRecord* jest adresem struktury definicji kanału (MQCD).

Jeśli nie ma następnego rekordu lub struktury, menedżer kolejek ustawia wartość *NextRecord* na pusty wskaźnik, a wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_NO_RECORD_AVAILABLE.

***CompCode* (MQLONG) -wyjście**

Kod zakończenia.

Kod zakończenia ma jedną z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

***Reason* (MQLONG) -wyjście**

Kod przyczyny kwalifikujący kod CompCode

Jeśli CompCode to MQCC_OK:

MQRC_NONE

(0, X'0000')

Brak powodu do zgłoszenia.

Jeśli CompCode to MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE

(2359, X'0937')

Brak dostępnych rekordów. Wywołanie MQXCLWLN zostało wysłane z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Bieżący rekord jest ostatnim rekordem w łańcuchu. Działanie naprawcze: Brak.

Jeśli CompCode to MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR

(2357, X'0935')

Parametr *CurrentRecord* jest niepoprawny. Wywołanie MQXCLWLN zostało wysłane z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Adres podany w parametrze *CurrentRecord* nie jest adresem poprawnego rekordu.

CurrentRecord musi być adresem rekordu docelowego, MQWDR, rekordu kolejki (MQWQR) lub rekordu klastra (MQWCR). rezydujących w pamięci podręcznej klastra. Czynność naprawczy:

upewnij się, że wyjście obciążenia klastra przekazuje adres poprawnego rekordu znajdującego się w pamięci podręcznej klastra.

**MQRC_ENVIRONMENT_ERROR
(2012, X'07DC')**

Wywołanie nie jest poprawne w środowisku. Wywołano komendę MQXCLWLN, ale nie z poziomu wyjścia obciążenia klastra.

**MQRC_NEXT_OFFSET_ERROR
(2358, X'0936')**

Parametr *NextOffset* jest niepoprawny. Wywołanie MQXCLWLN zostało wysłane z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Przesunięcie określone przez parametr *NextOffset* nie jest poprawne. *NextOffset* musi być wartością z jednego z następujących pól:

- Pole ChannelDefPrzesunięcie w Komponent MQWDR
- Pole ClusterRecOffset w obszarze MQWDR
- Pole ClusterRecOffset w obszarze MQWQR
- Pole ClusterRecOffset w MQWCR

Czynność naprawczy: należy upewnić się, że wartość określona dla parametru *NextOffset* jest wartością jednego z pól wymienionych wcześniej.

**MQRC_NEXT_RECORD_ERROR
(2361, X'0939')**

Parametr *NextRecord* jest niepoprawny.

**MQRC_WXP_ERROR
(2356, X'0934')**

Struktura parametru wyjścia obciążenia jest niepoprawna. Wywołanie MQXCLWLN zostało wysłane z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Struktura parametru wyjścia obciążenia *ExitParms* nie jest poprawna, z jednego z następujących powodów:

- Wskaźnik parametru jest niepoprawny. Wykrywanie wskaźników parametrów, które nie są poprawne, nie zawsze jest możliwe; jeśli nie zostaną wykryte, pojawiają się nieprzewidywalne wyniki.
- Pole StrucId nie jest polem MQWXP_STRUC_ID.
- Pole Wersja nie jest polem MQWXP_VERSION_2.
- Pole Kontekst nie zawiera wartości przekazanej do wyjścia przez menedżera kolejek.

Czynność naprawczy: upewnij się, że parametr określony dla *ExitParms* jest strukturą MQWXP, która została przekazana do wyjścia po wywołaniu wyjścia.

Odsyłacze pokrewne

Uwagi dotyczące użycia dla rekordów MQXCLWLN-Nawigacja w rekordach obciążenia klastra

Za pomocą programu MQXCLWLN można przechodzić między rekordami klastra, nawet jeśli pamięć podręczna jest statyczna.

Informacje o wywołaniach języka produktu MQXCLWLN

Produkt MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Uwagi dotyczące użycia dla rekordów MQXCLWLN-Nawigacja w rekordach obciążenia klastra

Za pomocą programu MQXCLWLN można przechodzić między rekordami klastra, nawet jeśli pamięć podręczna jest statyczna.

Jeśli pamięć podręczna klastra jest dynamiczna, należy użyć wywołania MQXCLWLN w celu nawigowania po rekordach. Wyjście kończy się nieprawidłowo, jeśli do przechodzenia przez rekordy są używane proste arytmetyki wskaźnikowe i offsetowe.

Jeśli pamięć podręczna klastra jest statyczna, program MQXCLWLN nie musi być używany do przechodzenia przez rekordy. Zwykle jest używany produkt MQXCLWLN nawet wtedy, gdy pamięć podręczna jest statyczna. Następnie można zmienić pamięć podręczną klastra w taki sposób, aby była ona dynamiczna bez konieczności zmiany wyjścia obciążenia.

Odsyłacze pokrewne

Parametry dla MQXCLWLN -nawigowanie rekordów obciążenia klastra
[Opis parametrów w wywołaniu MQXCLWLN](#) .

Informacje o wywołaniach języka produktu MQXCLWLN
 Produkt MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Informacje o wywołaniach języka produktu MQXCLWLN

Produkt MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Wywołanie C

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Zadeklaruj parametry w następujący sposób:

```
typedef struct tagMQXCLWLN {
MQWXP    ExitParms;      /* Exit parameter block */
MQPTR    CurrentRecord; /* Address of current record*/
MQLONG   NextOffset;    /* Offset of next record */
MQPTR    NextRecord;    /* Address of next record or structure */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
}
```

Wywołanie High Level Assembler

```
CALL MQXCLWLN, (CLWLEXITPARMS, CURRENTRECORD, NEXTOFFSET, NEXTRECORD, COMPCODE, REASON)
```

Zadeklaruj parametry w następujący sposób:

```
CLWLEXITPARMS CMQWXP, Cluster workload exit parameter block
CURRENTRECORD CMQWDRA, Current record
NEXTOFFSET    DS F      Next offset
NEXTRECORD    DS F      Next record
COMPCODE      DS F      Completion code
REASON        DS F      Reason code qualifying COMPCODE
```

Odsyłacze pokrewne

Parametry dla MQXCLWLN -nawigowanie rekordów obciążenia klastra
[Opis parametrów w wywołaniu MQXCLWLN](#) .

Uwagi dotyczące użycia dla rekordów MQXCLWLN-Nawigacja w rekordach obciążenia klastra
 Za pomocą programu MQXCLWLN można przechodzić między rekordami klastra, nawet jeśli pamięć podręczna jest statyczna.

MQWXP -Struktura parametru wyjścia obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

Tabela 195. Pola w produkcie MQWXP		
Pole	Opis	Strona
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja

Tabela 195. Pola w produkcie MQWXP (kontynuacja)

Pole	Opis	Strona
<i>ExitId</i>	Typ wyjścia	ExitId
<i>ExitReason</i>	Przyczyna wywołania wyjścia	ExitReason
<i>ExitResponse</i>	Odpowiedź z wyjścia	ExitResponse
<i>ExitResponse2</i>	Odpowiedź dodatkowa z wyjścia	ExitResponse2
<i>Feedback</i>	Kod zwrotny	Opinia
<i>Flags</i>	Wartości flag. Te znaczniki bitowe są używane do wskazania informacji na temat umieszczanego komunikatu.	Flagi
<i>ExitUserArea</i>	Wyjdź z obszaru użytkownika	ObszarExitUser
<i>ExitData</i>	Dane wyjścia	ExitData
<i>MsgDescPtr</i>	Adres deskryptora komunikatu (MQMD)	PtrMsgDesc
<i>MsgBufferPtr</i>	Adres buforu zawierającego niektóre lub wszystkie dane komunikatu	MsgBufferPtr
<i>MsgBufferLength</i>	Długość buforu zawierającego dane komunikatu	MsgBufferDługość
<i>MsgLength</i>	Długość kompletnego komunikatu	MsgLength
<i>QName</i>	Nazwa kolejki	Nazwa QName
<i>QMgrName</i>	Nazwa lokalnego menedżera kolejek	QMgrName
<i>DestinationCount</i>	Liczba możliwych miejsc docelowych	DestinationCount
<i>DestinationChosen</i>	Wybrane miejsce docelowe	DestinationChosen
<i>DestinationArrayPtr</i>	Adres tablicy wskaźników do rekordów docelowych (MQWDR)	DestinationArrayPtr
<i>QArrayPtr</i>	Adres tablicy wskaźników do rekordów kolejek (MQWQR)	QArrayPtr
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_2.		
<i>CacheContext</i>	Informacje o kontekście	CacheContext
<i>CacheType</i>	Typ pamięci podręcznej klastra	CacheType
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_3.		
<i>CLWLMRUChannels</i>	Maksymalna liczba dozwolonych aktywnych kanałów klastra wychodzącego	CLWLMRUChannels
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_4.		
<i>pEntryPoints</i>	Adres struktury MQIEP w celu umożliwienia wykonania wywołań MQI i DCI	pEntryPunkty

Struktura parametru wyjścia obciążenia klastra opisuje informacje, które są przekazywane do wyjścia obciążenia klastra.

Struktura parametru wyjścia obciążenia klastra jest obsługiwana na wszystkich platformach.

Ponadto w celu zapewnienia kompatybilności wstecznej dostępne są struktury MQWXP1, MQWXP2 i MQWXP3 .

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQXCLWLN -Nawiguj rekordy obciążenia klastra

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

MQWDR-Struktura rekordu miejsca docelowego obciążenia klastra

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

Pola w produkcji MQWXP -struktura parametru wyjścia obciążenia klastra

Opis pól w strukturze parametru wyjścia obciążenia klastra MQWXP

StrucId (MQCHAR4) -wejście

Identyfikator struktury dla struktury parametru wyjścia obciążenia klastra.

- Wartość StrucId to MQWXP_STRUC_ID.
- W przypadku języka programowania C jest również zdefiniowana stała MQWXP_STRUC_ID_ARRAY . Ma ona taką samą wartość jak MQWXP_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG) -wejście

Wskazuje numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWXP_VERSION_1

Struktura parametru wyjścia obciążenia klastra Version-1 .

Produkt MQWXP_VERSION_1 jest obsługiwany we wszystkich środowiskach.

MQWXP_VERSION_2

Struktura parametru wyjścia obciążenia klastra Version-2 .

Produkt MQWXP_VERSION_2 jest obsługiwany w następujących środowiskach: AIX, HP-UX, Linux, IBM i, Solaris i Windows.

MQWXP_VERSION_3

Struktura parametru wyjścia obciążenia klastra Version-3 .

Produkt MQWXP_VERSION_3 jest obsługiwany w następujących środowiskach: AIX, HP-UX, Linux, IBM i, Solaris i Windows.

MQWXP_VERSION_4

Struktura parametru wyjścia obciążenia klastra Version-4 .

Produkt MQWXP_VERSION_4 jest obsługiwany w następujących środowiskach: AIX, HP-UX, Linux, IBM i, Solaris i Windows.

MQWXP_CURRENT_VERSION

Bieżąca wersja struktury parametru wyjścia obciążenia klastra.

ExitId (MQLONG) -wejście

Wskazuje typ wywołanego wyjścia. Wyjście obciążenia klastra jest jedynym obsługiwany wyjściem.

- Wartość ExitId musi mieć wartość MQXT_CLUSTER_WORKLOAD_EXIT .

ExitReason (MQLONG) -wejście

Wskazuje przyczynę wywołania wyjścia obciążenia klastra. Parametr ExitReason przyjmuje jedną z następujących wartości:

MQXR_INIT

Wskazuje, że wyjście jest wywoływane po raz pierwszy.

Uzyskaj i zainicjuj wszystkie zasoby, które mogą być potrzebne dla wyjścia, takie jak pamięć główna.

MQXR_TERM

Wskazuje, że wyjście ma zostać zakończone.

Zwolnij wszystkie zasoby, które mogą zostać pozyskane od momentu jego zainicjowania, takie jak pamięć główna.

MQXR_CLWL_OPEN

Wywoływana przez produkt MQOPEN.

MQXR_CLWL_PUT

Wywoływana przez produkt MQPUT lub MQPUT1.

MQXR_CLWL_MOVE

Wywoływana przez MCA, gdy stan kanału uległ zmianie.

MQXR_CLWL_REPOS

Wywoływana przez produkt MQPUT lub MQPUT1 w przypadku komunikatu PCF menedżera repozytorium.

MQXR_CLWL_REPOS_MOVE

Wywoływana przez agenta MCA dla komunikatu PCF menedżera repozytorium, jeśli stan kanału został zmieniony.

ExitResponse (MQLONG) -wyjście

Ustaw wartość ExitResponse , aby określić, czy przetwarzanie komunikatu jest kontynuowane. Musi to być jedna z następujących wartości:

MQXCC_OK

Kontynuuj przetwarzanie komunikatu normalnie.

- DestinationChosen identyfikuje miejsce docelowe, do którego komunikat ma zostać wysłany.

MQXCC_SUPPRESS_FUNCTION

Przerwa przetwarzanie komunikatu.

- Działania podejmowane przez menedżera kolejek zależą od przyczyny, dla której wywołano wyjście:

Tabela 196. Działania podejmowane przez menedżer kolejek.

W tej tabeli znajdują się dwie kolumny. Pierwsza kolumna zawiera listę przyczyn wyjścia, a druga-opis działania podjętego dla każdej przyczyny wyjścia.

ExitReason	Wykonana czynność
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	Wywołanie MQOPEN, MQPUT lub MQPUT1 nie powiodło się z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_STOPPED_BY_CLUSTER_EXIT.
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	Komunikat jest umieszczany w kolejce niedostarczonych komunikatów.

MQXCC_SUPPRESS_EXIT

Kontynuuj przetwarzanie bieżącego komunikatu normalnie. Nie wywołuj ponownie wyjścia, dopóki menedżer kolejek nie zostanie wyłączony.

Menedżer kolejek przetwarza kolejne komunikaty tak, jakby atrybut menedżera kolejek ClusterWorkloadExit był pusty. DestinationChosen identyfikuje miejsce docelowe, do którego wysłany jest bieżący komunikat.

Każda inna wartość

Przetwórz komunikat tak, jakby został podany MQXCC_SUPPRESS_FUNCTION .

ExitResponse2 (MQLONG) -input/output

Ustaw wartość ExitResponse2 , aby udostępnić menedżerowi kolejek więcej informacji.

- MQXR2_STATIC_CACHE jest wartością domyślną i jest ustawiana przy wejściu do wyjścia.
- Jeśli parametr ExitReason ma wartość MQXR_INIT, wyjście może ustawić jedną z następujących wartości w polu ExitResponse2:

MQXR2_STATIC_CACHE

Wyjście wymaga pamięci podręcznej klastra statycznego.

- Jeśli pamięć podręczna klastra jest statyczna, wyjście nie musi używać wywołania MQXCLWLN do nawigowania w łańcuchach rekordów w pamięci podręcznej klastra.
- Jeśli pamięć podręczna klastra jest dynamiczna, wyjście nie może być poprawnie nawigowane przez rekordy w pamięci podręcznej.

Uwaga: Menedżer kolejek przetwarza zwrot z wywołania MQXR_INIT tak, jakby to wyjście zwróciło MQXCC_SUPPRESS_EXIT w polu ExitResponse .

MQXR2_DYNAMIC_CACHE

Wyjście może działać z pamięcią podręczną statyczną lub dynamiczną.

- Jeśli wyjście zwraca tę wartość, wyjście musi używać wywołania MQXCLWLN w celu nawigowania w łańcuchach rekordów w pamięci podręcznej klastra.

Opinia (MQLONG) -wejście

Pole zarezerwowane. Wartość wynosi zero.

Flags (MQLONG) -wejście

Wskazuje informacje na temat umieszczanego komunikatu.

- Wartością opcji Flags jest MQWXP_PUT_BY_CLUSTER_CHL. Komunikat pochodzi z kanału klastra, a nie lokalnie lub z kanału innego niż kanał klastra. Innymi słowy, komunikat pochodzi z innego menedżera kolejek klastra.

Zastrzeżone (MQLONG) -wejście

Pole zarezerwowane. Wartość wynosi zero.

ObszarExitUser (MQBYTE16) -input/output

Ustaw opcję ExitUserArea , aby komunikować się między wywołaniami do wyjścia.

- Program ExitUserObszar jest inicjowany do zera binarnego przed pierwszym wywołaniem wyjścia. Wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane w wywołaniach wyjścia, które występują między wywołaniem MQCONN i pasującym wywołaniem MQDISC . Podczas wywołania MQDISC pole jest resetowane do zera binarnego.
- Pierwsze wywołanie wyjścia jest wskazywane przez pole ExitReason o wartości MQXR_INIT.
- Zdefiniowane są następujące stałe:

MQXUA_NONE -łańcuch

MQXUA_NONE_ARRAY -tablica znakowa

Brak informacji o użytkowniku. Obie stałe są binarne zero dla długości pola.

MQ_EXIT_USER_AREA_LENGTH

Długość obszaru ExitUserArea.

ExitData (MQCHAR32) -wejście

Wartość atrybutu menedżera kolejek ClusterWorkloadData . Jeśli dla tego atrybutu nie zdefiniowano żadnej wartości, pole jest puste.

- Długość parametru ExitData jest podawana przez produkt MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD) -wejście

Adres kopii deskryptora komunikatu (MQMD) dla przetwarzanego komunikatu.

- Wszystkie zmiany wprowadzone w deskrytorze komunikatu przez wyjście są ignorowane przez menedżer kolejek.
- Jeśli parametr `ExitReason` ma jedną z następujących wartości `MsgDescPtr` , jest ustawiony na pusty wskaźnik, a do wyjścia nie jest przekazywany żaden deskryptor komunikatu:
 - `MQXR_INIT`
 - `MQXR_TERM`
 - `MQXR_CLWL_OPEN`

MsgBufferPtr (PMQVOID) -wejście

Adres buforu, który zawiera kopię pierwszych bajtów `MsgBuffer` danych komunikatu.

- Wszystkie zmiany wprowadzone w danych komunikatu przez wyjście są ignorowane przez menedżer kolejek.
- Żadne dane komunikatu nie są przekazywane do wyjścia, gdy:
 - `MsgDescPtr` -wskaźnik pusty.
 - Komunikat nie zawiera danych.
 - Atrybut menedżera kolejek `ClusterWorkloadLength` ma wartość zero.

W takich przypadkach parametr `MsgBufferPtr` jest wskaźnikiem wartości `NULL`.

MsgBufferLength (MQLONG) -wejście

Długość buforu zawierającego dane komunikatu przekazywane do wyjścia.

- Długość jest kontrolowana za pomocą atrybutu menedżera kolejek `ClusterWorkloadLength` .
- Długość może być mniejsza niż długość kompletnego komunikatu, patrz `MsgLength`.

MsgLength (MQLONG) -wejście

Długość kompletnego komunikatu przekazanego do wyjścia.

- Wartość `MsgBufferLength` może być mniejsza niż długość kompletnego komunikatu.
- Wartość `MsgLength` wynosi zero, jeśli `ExitReason` to `MQXR_INIT`, `MQXR_TERM` lub `MQXR_CLWL_OPEN`.

QName (MQCHAR48) -wejście

Nazwa kolejki docelowej. Kolejka jest kolejką klastra.

- Długość nazwy `QName` to `MQ_Q_NAME_LENGTH`.

QMgrName (MQCHAR48) -wejście

Nazwa lokalnego menedżera kolejek, który wywołał wyjście obciążenia klastra.

- Długość parametru `QMgrName` to `MQ_Q_MGR_NAME_LENGTH`.

DestinationCount (MQLONG) -wejście

Liczba możliwych miejsc docelowych. Miejsca docelowe są instancjami kolejki docelowej i są opisywane przez rekordy miejsca docelowego.

- Rekord docelowy jest strukturą `MQWDR` . Istnieje jedna struktura dla każdej możliwej trasy do każdej instancji kolejki.
- Struktury produktu `MQWDR` są adresowane przez tablicę wskaźników, patrz sekcja `DestinationArrayPtr`.

DestinationChosen (MQLONG) -input/output

Wybrane miejsce docelowe.

- Numer struktury `MQWDR` , która identyfikuje trasę i instancję kolejki, do której komunikat ma zostać wysłany.
- Wartość znajduje się w zakresie 1- `DestinationCount`.

- Po wejściu do wyjścia opcja `DestinationChosen` wskazuje trasę i instancję kolejki, która została wybrana przez menedżer kolejek. Wyjście może zaakceptować ten wybór lub wybrać inną trasę i instancję kolejki.
- Wartość ustawiona przez wyjście musi mieścić się w zakresie 1- `DestinationCount`. Jeśli zostanie zwrócona dowolna inna wartość, menedżer kolejek użyje wartości `DestinationChosen` na wejściu do wyjścia.

DestinationArrayPtr (PPMQWDR) -wejście

Adres tablicy wskaźników do rekordów docelowych (MQWDR).

- Istnieją rekordy miejsca docelowego `DestinationCount` .

QArrayPtr (PPMQWQR) -wejście

Adres tablicy wskaźników do rekordów kolejek (MQWQR).

- Jeśli dostępne są rekordy kolejek, dla nich znajduje się wartość `DestinationCount` .
- Jeśli nie są dostępne żadne rekordy kolejek, parametr `QArrayPtr` jest wskaźnikiem wartości NULL.

Uwaga: `QArrayPtr` może być wskaźnikiem wartości NULL, nawet jeśli wartość `DestinationCount` jest większa od zera.

CacheContext (MQPTR) : Wersja 2-wejście

Pole `CacheContext` jest zarezerwowane do użycia przez menedżer kolejek. Wyjście nie może zmieniać wartości tego pola.

CacheType (MQLONG) : Wersja 2-wejście

Pamięć podręczna klastra ma jeden z następujących typów:

MQCLCT_STATIC

Pamięć podręczna jest statyczna.

- Wielkość pamięci podręcznej jest stała i nie może rosnąć w czasie działania menedżera kolejek.
- Nie ma potrzeby używania wywołania `MQXCLWLN` do nawigowania w rekordach w tym typie pamięci podręcznej.

MQCLCT_DYNAMIC

Pamięć podręczna jest dynamiczna.

- Wielkość pamięci podręcznej może być zwiększana w celu uwzględnienia różnych informacji o klastrze.
- Konieczne jest użycie wywołania `MQXCLWLN` do nawigowania w rekordach w tym typie pamięci podręcznej.

CLWLMRUChannel (MQLONG) : Wersja 3-wejście

Wskazuje maksymalną liczbę aktywnych wychodzących kanałów klastra, które mają być uwzględnione w algorytmie wyboru obciążenia klastra.

- `CLWLMRUChannel` jest wartością z zakresu od 1 do 999 999 999.

pEntryPunkty (PMQIEP) : Wersja 4

Adres struktury `MQIEP`, za pośrednictwem której mogą być wykonywane wywołania `MQI` i `DCI`.

Odsyłacze pokrewne

Wartości początkowe i deklaracje języków dla produktu `MQWXP`

Wartości początkowe i deklaracje języka C i High Level Assembler dla struktury parametru wyjścia obciążenia klastra `MQWXP` .

Wartości początkowe i deklaracje języków dla produktu MQWXP

Wartości początkowe i deklaracje języka C i High Level Assembler dla struktury parametru wyjścia obciążenia klastra `MQWXP` .

Tabela 197. Początkowe wartości pól w produkcji MQWXP

Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQWXP_STRUC_ID	'WXP¬'
<i>Version</i>	MQWXP_VERSION_2	2
<i>ExitId</i>	Brak	0
<i>ExitReason</i>	MQXCC_OK	0
<i>ExitResponse</i>	Brak	0
<i>ExitResponse2</i>	Brak	0
<i>Flags</i>	Brak	0
<i>ExitUserArea</i>	{MQXUA_NONE_ARRAY}	0
<i>ExitData</i>	Brak	" "
<i>MsgDescPtr</i>	Brak	NULL
<i>MsgBufferPtr</i>	Brak	NULL
<i>MsgBufferLength</i>	Brak	0
<i>MsgBufferPtr</i>	Brak	0
<i>QName</i>	Brak	" "
<i>QMgrName</i>	Brak	" "
<i>DestinationCount</i>	Brak	0
<i>DestinationChosen</i>	Brak	0
<i>DestinationArrayPtr</i>	Brak	NULL
<i>QArrayPtr</i>	Brak	NULL
<i>CacheContext</i>	Brak	NULL
<i>CacheType</i>	MQCLCT_DYNAMIC	1
<i>CLWLMRUChannels</i>	Brak	0
<i>pEntryPoints</i>	Brak	NULL

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy pusty znak.
2. W języku programowania C zmienna makra MQWXP_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQWXP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     ExitId;           /* Type of exit */
    MQLONG     ExitReason;       /* Reason for invoking exit */
    MQLONG     ExitResponse;     /* Response from exit */
}
```

```

MQLONG    ExitResponse2;      /* Reserved */
MQLONG    Feedback;          /* Reserved */
MQLONG    Flags;              /* Flags */
MQBYTE16  ExitUserArea;      /* Exit user area */
MQCHAR32  ExitData;          /* Exit data */
PMQMD     MsgDescPtr;        /* Address of message descriptor */
PMQVOID    MsgBufferPtr;      /* Address of buffer containing some
                               or all of the message data */
MQLONG    MsgBufferLength;    /* Length of buffer containing message
                               data */
MQLONG    MsgLength;         /* Length of complete message */
MQCHAR48  QName;             /* Queue name */
MQCHAR48  QMgrName;          /* Name of local queue manager */
MQLONG    DestinationCount;   /* Number of possible destinations */
MQLONG    DestinationChosen;  /* Destination chosen */
PPMQWDR   DestinationArrayPtr; /* Address of an array of pointers to
                               destination records */
PPMQWQR   QArrayPtr;         /* Address of an array of pointers to
                               queue records */

/* version 1 */
MQPTR     CacheContext;      /* Context information */
MQLONG    CacheType;         /* Type of cluster cache */
/* version 2 */
MQLONG    CLWLMRUChannels;    /* Maximum number of most recently
                               used cluster channels */

/* version 3 */
PMQIEP    pEntryPoints;      /* Address of the MQIEP structure */
/* version 4 */
};

```

High Level Assembler

MQWXP	DSECT		
MQWXP_STRUCID	DS	CL4	Structure identifier
MQWXP_VERSION	DS	F	Structure version number
MQWXP_EXITID	DS	F	Type of exit
MQWXP_EXITREASON	DS	F	Reason for invoking exit
MQWXP_EXITRESPONSE	DS	F	Response from exit
MQWXP_EXITRESPONSE2	DS	F	Reserved
MQWXP_FEEDBACK	DS	F	Reserved
MQWXP_RESERVED	DS	F	Reserved
MQWXP_EXITUSERAREA	DS	XL16	Exit user area
MQWXP_EXITDATA	DS	CL32	Exit data
MQWXP_MSGDESCPTR	DS	F	Address of message descriptor
* MQWXP_MSGBUFFERPTR	DS	F	Address of buffer containing some or all of the message data
* MQWXP_MSGBUFFERLENGTH	DS	F	Length of buffer containing message data
* MQWXP_MSGLENGTH	DS	F	Length of complete message
MQWXP_QNAME	DS	CL48	Queue name
MQWXP_QMGRNAME	DS	CL48	Name of local queue manager
MQWXP_DESTINATIONCOUNT	DS	F	Number of possible destinations
* MQWXP_DESTINATIONCHOSEN	DS	F	Destination chosen
MQWXP_DESTINATIONARRAYPTR	DS	F	Address of an array of pointers to destination records
* MQWXP_QARRAYPTR	DS	F	Address of an array of pointers to queue records
* MQWXP_CACHECONTEXT	DS	F	Context information
MQWXP_CACHETYPE	DS	F	Type of cluster cache
MQWXP_CLWLMRUCHANNELS	DS	F	Number of most recently used channels for workload balancing
* MQWXP_LENGTH	EQU	*-MQWXP	Length of structure
MQWXP_AREA	ORG	MQWXP	
	DS	CL(MQWXP_LENGTH)	

Odnośniki pokrewne

[Pola w produkcie MQWXP -struktura parametru wyjścia obciążenia klastra](#)

[Opis pól w strukturze parametru wyjścia obciążenia klastra MQWXP](#)

MQWDR-Struktura rekordu miejsca docelowego obciążenia klastra

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

Tabela 198. Pola w MQWDR		
Pole	Opis	Strona
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury MQWDR	StrucLength
<i>QMgrFlags</i>	Flagi menedżera kolejek	QMgrFlags
<i>QMgrIdentifier</i>	Identyfikator menedżera kolejek	QMgrIdentifier
<i>QMgrName</i>	Nazwa menedżera kolejek	QMgrName
<i>ClusterRecOffset</i>	Przesunięcie logiczne pierwszego rekordu klastra (MQWCR)	ClusterRecPrzesunięcie
<i>ChannelState</i>	Stan kanału	ChannelState
<i>ChannelDefOffset</i>	Przesunięcie logiczne struktury definicji kanału (MQCD)	ChannelDefPrzesunięcie
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWDR_VERSION_2.		
<i>DestSeqNumber</i>	Numer kolejny miejsca docelowego kanału	DestSeq
<i>DestSeqFactor</i>	Współczynnik sekwencji miejsca docelowego kanału dla ważenia	DestSeqWspółczynnik

Struktura rekordu miejsca docelowego obciążenia klastra zawiera informacje odnoszące się do jednego z możliwych miejsc docelowych dla komunikatu. Dla każdej instancji kolejki docelowej istnieje jedna struktura rekordu miejsca docelowego obciążenia klastra.

Struktura rekordu miejsca docelowego obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Ponadto w celu zapewnienia kompatybilności wstecznej dostępne są struktury MQWDR1 i MQWDR2 .

Odsyłacze pokrewne

[MQ_CLUSTER_WORKLOAD_EXIT](#) -opis połączenia

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

[MQXCLWLN](#) -Nawiguj rekordy obciążenia klastra

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

[MQWXP](#) -Struktura parametru wyjścia obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

[MQWQR](#) -Struktura rekordu kolejki obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

[MQWCR](#) -Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

Pola w strukturze rekordu docelowego obciążenia klastra-MQWDR-Cluster

Opis parametrów w strukturze rekordu docelowego obciążenia klastra MQWDR .

StrucId (MQCHAR4) -wejście

Identyfikator struktury dla struktury rekordu miejsca docelowego obciążenia klastra.

- Wartość StrucId to MQWDR_STRUC_ID.
- W przypadku języka programowania C jest również zdefiniowana stała MQWDR_STRUC_ID_ARRAY . Ma ona taką samą wartość jak MQWDR_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG) -wejście

Numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWDR_VERSION_1

Rekord miejsca docelowego obciążenia klastra Version-1 .

MQWDR_VERSION_2

Rekord miejsca docelowego obciążenia klastra Version-2 .

MQWDR_CURRENT_VERSION

Bieżąca wersja rekordu miejsca docelowego obciążenia klastra.

StrucLength (MQLONG) -wejście

Długość struktury MQWDR . StrucLength przyjmuje jedną z następujących wartości:

MQWDR_LENGTH_1

Długość rekordu miejsca docelowego obciążenia klastra w wersji version-1 .

MQWDR_LENGTH_2

Długość rekordu miejsca docelowego obciążenia klastra w wersji version-2 .

MQWDR_CURRENT_LENGTH

Długość bieżącej wersji rekordu miejsca docelowego obciążenia klastra.

QMgrFlags (MQLONG) -wejście

Flagi menedżera kolejek wskazujące właściwości menedżera kolejek, które udostępnia instancję kolejki docelowej opisanej w strukturze MQWDR . Zdefiniowane są następujące opcje:

MQQMF_REPOSITORY_Q_MGR

Miejsce docelowe jest pełnym menedżerem kolejek repozytorium.

MQQMF_CLUSSDR_USER_DEFINED

Kanał wysyłający klastry został zdefiniowany ręcznie.

MQQMF_CLUSSDR_AUTO_DEFINED

Kanał wysyłający klastry został zdefiniowany automatycznie.

MQQMF_AVAILABLE

Docelowy menedżer kolejek jest dostępny do odbierania komunikatów.

Inne wartości

Inne flagi w tym polu mogą być ustawione przez menedżer kolejek do celów wewnętrznych.

QMgrIdentifier (MQCHAR48) -wejście

Identyfikator menedżera kolejek jest unikalnym identyfikatorem menedżera kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWDR .

- Identyfikator jest generowany przez menedżer kolejek.
- Długość parametru QMgrIdentifier to MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) -wejście

Nazwa menedżera kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWDR .

- QMgrName może być nazwą lokalnego menedżera kolejek, jak również innego menedżera kolejek w klastrze.
- Długość parametru QMgrName to MQ_Q_MGR_NAME_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście

Przesunięcie logiczne pierwszej struktury MQWCR , która należy do struktury MQWDR .

- W przypadku pamięci podręcznych statycznych pozycja ClusterRecOffset jest przesuniętą pierwszą strukturą MQWCR , która należy do struktury MQWDR .
- Przesunięcie jest mierzone w bajtach od początku struktury MQWDR .

- Nie należy używać przesunięcia logicznego dla arytmetyki wskaźnika z dynamicznymi pamięciami podręcznymi. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

ChannelState (MQLONG) -wejście

Stan kanału, który łączy lokalny menedżer kolejek z menedżerem kolejek identyfikowany przez strukturę MQWDR . Dozwolone są następujące wartości:

MQCHS_BINDING

Kanał negocjuje z partnerem.

MQCHS_INACTIVE

Kanał nie jest aktywny.

MQCHS_INITIALIZING

Kanał jest inicjowany.

MQCHS_PAUSED

Kanał został wstrzymany.

MQCHS_REQUESTING

Kanał requestera żąda połączenia.

MQCHS_RETRYING

Kanał próbuje nawiązać połączenie.

MQCHS_RUNNING

Kanał przesyła lub oczekuje na komunikaty.

MQCHS_STARTING

Kanał oczekuje na aktywne działanie.

MQCHS_STOPPING

Kanał jest zatrzymywany.

MQCHS_STOPPED

Kanał został zatrzymany.

ChannelDefPrzesunięcie (MQLONG) -wejście

Przesunięcie logiczne definicji kanału (MQCD) Dla kanału, który łączy lokalny menedżer kolejek z menedżerem kolejek identyfikowany przez strukturę MQWDR .

- ChannelDefPrzesunięcie jest podobne do pozycji ClusterRecOffset
- Przesunięcie logiczne nie może być używane w arytmetycznym wskaźniku. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

DestSeqWspółczynnik (MQLONG) -wejście

Docelowy czynnik sekwencji, który umożliwia wybór kanału w oparciu o wagę.

- Element DestSeqFactor jest używany przed zmianami w menedżerze kolejek.
- Menedżer obciążenia zwiększa współczynnik DestSeqFactor w taki sposób, aby komunikaty były dystrybuowane w dół przez kanały w zależności od ich wagi.

DestSeqNumer (MQLONG) -wejście

Wartość miejsca docelowego kanału klastra, zanim menedżer kolejek je zmieni.

- Menedżer obciążenia zwiększa wartość DestSeqNumber za każdym razem, gdy komunikat jest umieszczany w tym kanale.
- Wyjścia obciążenia mogą używać parametru DestSeqNumber , aby zdecydować, który kanał ma zostać umieszczony w dół.

Odsyłacze pokrewne

Wartości początkowe i deklaracje języka dla produktu MQWDR

Wartości początkowe i deklaracje języka C i High Level Assembler dla rekordu miejsca docelowego obciążenia klastra MQWDR .

Wartości początkowe i deklaracje języka dla produktu MQWDR

Wartości początkowe i deklaracje języka C i High Level Assembler dla rekordu miejsca docelowego obciążenia klastra MQWDR .

Tabela 199. Początkowe wartości pól w produkcie MQWDR		
Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucId</i>	MQWDR_STRUC_ID	'WDR~'
<i>Version</i>	MQWDR_VERSION_1	1
<i>StrucLength</i>	MQWDR_CURRENT_LENGTH ³	136
<i>QMgrFlags</i>	MQWDR_NONE	0
<i>QMgrIdentifier</i>	Brak	" "
<i>QMgrName</i>	Brak	" "
<i>ClusterRecOffset</i>	Brak	0
<i>ChannelState</i>	Brak	0
<i>ChannelDefOffset</i>	Brak	0
<i>DestSeqNumber</i>	Brak	0
<i>DestSeqFactor</i>	Brak	0
Uwagi: - Symbol ~ reprezentuje pojedynczy pusty znak. - W języku programowania C zmienna makra MQWDR_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze: <pre>MQWDR MyWDR = {MQWDR_DEFAULT};</pre> - Wartości początkowe celowo ustawiają długość struktury na długość bieżącej wersji, a nie w wersji 1 struktury.		

High Level Assembler

```

MQWDR
MQWDR_STRUCID      DS    CL4      Structure identifier
MQWDR_VERSION      DS    F        Structure version number
MQWDR_STRUCLNGTH   DS    F        Length of MQWDR structure
MQWDR_QMGRFLAGS    DS    F        Queue-manager flags
MQWDR_QMGRIDENTIFIER DS    CL48   Queue-manager identifier
MQWDR_QMGRNAME     DS    CL48   Queue-manager name
MQWDR_CLUSTERRECOFFSET DS    F    Offset of first cluster
*                  record
MQWDR_CHANNELSTATE DS    F        Channel state
MQWDR_CHANNELDEFOFFSET DS    F    Offset of channel definition
*                  structure
MQWDR_LENGTH       EQU    *-MQWDR Length of structure
MQWDR_AREA         ORG    MQWDR
                   DS     CL(MQWDR_LENGTH)
```

Deklaracja C

```

typedef struct tagMQWDR {
    MQCHAR4    StrucId;          /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;      /* Length of MQWDR structure */
}
```



```

MQLONG    QMgrFlags;          /* Queue-manager flags */
MQCHAR48  QMgrIdentifier;     /* Queue-manager identifier */
MQCHAR48  QMgrName;          /* Queue-manager name */
MQLONG    ClusterRecOffset;   /* Offset of first cluster record */
MQLONG    ChannelState;      /* Channel state */
MQLONG    ChannelDefOffset;   /* Offset of channel definition structure */
/* Ver:1 */
MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
MQINT64   DestSeqFactor;     /* Cluster channel factor sequence number */
/* Ver:2 */
};

```

Odsyłacze pokrewne

Pola w strukturze rekordu docelowego obciążenia klastra-MQWDR-Cluster

Opis parametrów w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

Tabela 200. Pola w obszarze MQWQR		
Pole	Opis	Strona
<i>StrucId</i>	Identyfikator struktury	StrucId
<i>Version</i>	Numer wersji struktury	Wersja
<i>StrucLength</i>	Długość struktury produktu MQWQR	StrucLength
<i>QFlags</i>	Flagi kolejki	QFlags
<i>QName</i>	Nazwa kolejki	Nazwa QName
<i>QMgrIdentifier</i>	Identyfikator menedżera kolejek	QMgrIdentifier
<i>ClusterRecOffset</i>	Przesunięcie pierwszego rekordu klastra (MQWCR)	ClusterRecPrzesunięcie
<i>QType</i>	Typ kolejki	QTYPE
<i>QDesc</i>	Opis kolejki	QDesc
<i>DefBind</i>	Domyślne łączenie	DefBind
<i>DefPersistence</i>	Domyślna trwałość komunikatu	DefPersistence
<i>DefPriority</i>	Domyślny priorytet komunikatu	DefPriority
<i>InhibitPut</i>	Czy dozwolone są operacje umieszczania w kolejce	InhibitPut
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWQR_VERSION_2.		
<i>CWLQueuePriority</i>	Wartość 0-9 reprezentująca priorytet kolejki.	CLWLQueuePriority
<i>CLWLQueueRank</i>	Wartość 0-9 reprezentująca rangę kolejki	CLWLQueueRank
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWQR_VERSION_3.		
<i>DefPutResponse</i>	Operacja put - domyślna odp.	OdpowiedźDefPut

Struktura rekordu kolejki obciążenia klastra zawiera informacje odnoszące się do jednego z możliwych miejsc docelowych dla komunikatu. Dla każdej instancji kolejki docelowej istnieje jedna struktura rekordu kolejki obciążenia klastra.

Struktura rekordu kolejki obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Ponadto w celu zapewnienia kompatybilności wstecznej dostępne są struktury MQWQR1 i MQWQR2 .

Odsyłacze pokrewne

[MQ_CLUSTER_WORKLOAD_EXIT](#) - opis połączenia

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

[MQXCLWLN](#) - Nawiguj rekordy obciążenia klastra

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

[MQWXP](#) - Struktura parametru wyjścia obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

[MQWDR](#) - Struktura rekordu miejsca docelowego obciążenia klastra

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

[MQWCR](#) - Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

Pola w tabeli MQWQR -Struktura rekordu kolejki obciążenia klastra

Opis pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

StrucId (MQCHAR4) -wejście

Identyfikator struktury dla struktury rekordu kolejki obciążenia klastra.

- Wartość StrucId to MQWQR_STRUC_ID.
- W przypadku języka programowania C jest również zdefiniowana stała MQWQR_STRUC_ID_ARRAY . Ma ona taką samą wartość jak MQWQR_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG) -wejście

Numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWQR_VERSION_1

Rekord kolejki obciążenia klastra Version-1 .

MQWQR_VERSION_2

Rekord kolejki obciążenia klastra Version-2 .

MQWQR_VERSION_3

Rekord kolejki obciążenia klastra Version-3 .

MQWQR_CURRENT_VERSION

Bieżąca wersja rekordu kolejki obciążenia klastra.

StrucLength (MQLONG) -wejście

Długość struktury MQWQR . StrucLength przyjmuje jedną z następujących wartości:

MQWQR_LENGTH_1

Długość rekordu kolejki obciążenia klastra w wersji version-1 .

MQWQR_LENGTH_2

Długość rekordu kolejki obciążenia klastra version-2 .

MQWQR_LENGTH_3

Długość rekordu kolejki obciążenia klastra version-3 .

MQWQR_CURRENT_LENGTH

Długość bieżącej wersji rekordu kolejki obciążenia klastra.

QFlags (MQLONG) -wejście

Flagi kolejki wskazują właściwości kolejki. Zdefiniowane są następujące opcje:

MQQF_LOCAL_Q

Miejsce docelowe jest kolejką lokalną.

MQQF_CLWL_USEQ_ANY

Zezwala na użycie kolejek lokalnych i zdalnych.

MQQF_CLWL_USEQ_LOCAL

Zezwala na umieszczanie tylko kolejki lokalnej.

Inne wartości

Inne flagi w tym polu mogą być ustawione przez menedżer kolejek do celów wewnętrznych.

QName (MQCHAR48) -wejście

Nazwa kolejki, która jest jednym z możliwych miejsc docelowych komunikatu.

- Długość nazwy QName to MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) -wejście

Identyfikator menedżera kolejek jest unikalnym identyfikatorem menedżera kolejek, który udostępnia instancję kolejki opisaną w strukturze MQWQR .

- Identyfikator jest generowany przez menedżer kolejek.
- Długość parametru QMgrIdentifier to MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście

Przesunięcie logiczne pierwszej struktury MQWCR , która należy do struktury MQWQR .

- W przypadku pamięci podręcznych statycznych pozycja ClusterRecOffset jest przesunięta pierwszą strukturą MQWCR , która należy do struktury MQWQR .
- Przesunięcie jest mierzone w bajtach od początku struktury MQWQR .
- Nie należy używać przesunięcia logicznego dla arytmetyki wskaźnika z dynamicznymi pamięciami podręcznymi. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

Typ QType (MQLONG) -wejście

Typ kolejki docelowej kolejki. Dozwolone są następujące wartości:

MQCQT_LOCAL_Q

Kolejka lokalna.

MQCQT_ALIAS_Q

Kolejka aliasowa.

MQCQT_REMOTE_Q

Kolejka zdalna.

MQCQT_Q_MGR_ALIAS

Alias menedżera kolejek.

QDesc (MQCHAR64) -wejście

Atrybut kolejki opisu kolejki zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR .

- Długość QDesc to MQ_Q_DESC_LENGTH.

DefBind (MQLONG) -wejście

Domyślny atrybut kolejki powiązania zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR . W przypadku używania grup z klastrami należy określić wartość MQBND_BIND_ON_OPEN lub MQBND_BIND_ON_GROUP . Możliwe są następujące wartości:

MQBND_BIND_ON_OPEN

Powiązanie ustalone przez wywołanie MQOPEN .

MQBND_BIND_NOT_FIXED

Powiązanie nie zostało ustalone.

MQBND_BIND_ON_GROUP

Umożliwia aplikacji żądanie, aby grupa komunikatów była przydzielona do tej samej instancji docelowej.

DefPersistence (MQLONG) -wejście

Domyślny atrybut kolejki trwałości komunikatów zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR . Dozwolone są następujące wartości:

MQPER_PERSISTENT

Komunikat jest trwały.

MQPER_NOT_PERSISTENT

Komunikat nie jest trwały.

DefPriority (MQLONG) -wejście

Domyślny atrybut kolejki priorytetów komunikatów zdefiniowany w menedżerze kolejek, który obsługuje instancję kolejki docelowej opisaną w strukturze MQWQR . Zakres priorytetu ma wartość 0- MaxPriority.

- Wartość 0 jest najniższym priorytetem.
- MaxPriority jest atrybutem menedżera kolejek menedżera kolejek, który udostępnia tę instancję kolejki docelowej.

InhibitPut (MQLONG) -wejście

Atrybut kolejki zablokowana zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR . Dozwolone są następujące wartości:

MQQA_PUT_INHIBITED

Operacje put są zablokowane.

MQQA_PUT_ALLOWED

Operacje put są dozwolone.

CLWLQueuePriority (MQLONG) -wejście

Atrybut priorytetu kolejki obciążenia klastra zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR .

CLWLQueueRank (MQLONG) -wejście

Obszar kolejki obciążenia klastra zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR .

OdpowiedźDefPut (MQLONG) -wejście

Domyślny atrybut kolejki umieszczania odpowiedzi zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisaną w strukturze MQWQR . Dozwolone są następujące wartości:

MQPRT_SYNC_RESPONSE

Synchroniczna odpowiedź na wywołania programu MQPUT lub MQPUT1 .

MQPRT_ASYNC_RESPONSE

Asynchroniczna odpowiedź na wywołania programu MQPUT lub MQPUT1 .

Odsyłacze pokrewne

Wartości początkowe i deklaracje języków dla produktu MQWQR

Wartości początkowe i deklaracje języka C i High Level Assembler dla rekordu kolejki obciążenia klastra MQWQR .

Wartości początkowe i deklaracje języków dla produktu MQWQR

Wartości początkowe i deklaracje języka C i High Level Assembler dla rekordu kolejki obciążenia klastra MQWQR .

Tabela 201. Wartości początkowe pól w MQWQR		
Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR~'
Version	MQWQR_VERSION_1	1

Tabela 201. Wartości początkowe pól w MQWQR (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej
<i>StrucLength</i>	MQWQR_CURRENT_LENGTH ³	212
<i>QFlags</i>	Brak	0
<i>QName</i>	Brak	" "
<i>QMgrIdentifier</i>	Brak	" "
<i>ClusterRecOffset</i>	Brak	0
<i>QType</i>	Brak	0
<i>QDesc</i>	Brak	" "
<i>DefBind</i>	Brak	0
<i>DefPersistence</i>	Brak	0
<i>DefPriority</i>	Brak	0
<i>InhibitPut</i>	Brak	0
<i>CLWLQueuePriority</i>	Brak	0
<i>CLWLQueueRank</i>	Brak	0
<i>DefPutResponse</i>	Brak	1

Uwagi:

1. Symbol – reprezentuje pojedynczy pusty znak.
2. W języku programowania C zmienna makra MQWQR_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać początkowe wartości dla pól w strukturze:

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. Wartości początkowe celowo ustawiają długość struktury na długość bieżącej wersji, a nie w wersji 1 struktury.

Deklaracja C

```
typedef struct tagMQWQR {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;           /* Structure version number */
    MQLONG     StrucLength;       /* Length of MQWQR structure */
    MQLONG     QFlags;            /* Queue flags */
    MQCHAR48   QName;             /* Queue name */
    MQCHAR48   QMgrIdentifier;     /* Queue-manager identifier */
    MQLONG     ClusterRecOffset;  /* Offset of first cluster record */
    MQLONG     QType;             /* Queue type */
    MQCHAR64   QDesc;             /* Queue description */
    MQLONG     DefBind;           /* Default binding */
    MQLONG     DefPersistence;    /* Default message persistence */
    MQLONG     DefPriority;       /* Default message priority */
    MQLONG     InhibitPut;        /* Whether put operations on the queue
                                are allowed */

    /* version 2 */
    MQLONG     CLWLQueuePriority; /* Queue priority */
    MQLONG     CLWLQueueRank;    /* Queue rank */
    /* version 3 */
    MQLONG     DefPutResponse;   /* Default put response */
};
```

High Level Assembler

MQWQR	DSECT		
MQWQR_STRUCID	DS	CL4	Structure identifier
MQWQR_VERSION	DS	F	Structure version number
MQWQR_STRUCLength	DS	F	Length of MQWQR structure
MQWQR_QFLAGS	DS	F	Queue flags
MQWQR_QNAME	DS	CL48	Queue name
MQWQR_QMGRIDENTIFIER	DS	CL48	Queue-manager identifier
MQWQR_CLUSTERRECOFFSET	DS	F	Offset of first cluster
*			record
MQWQR_QTYPE	DS	F	Queue type
MQWQR_QDESC	DS	CL64	Queue description
MQWQR_DEFBIND	DS	F	Default binding
MQWQR_DEFPERSISTENCE	DS	F	Default message persistence
MQWQR_DEFPRIORITY	DS	F	Default message priority
MQWQR_INHIBITPUT	DS	F	Whether put operations on
*			the queue are allowed
MQWQR_DEFPUTRESPONSE	DS	F	Default put response
MQWQR_LENGTH	EQU	*-MQWQR	Length of structure
	ORG	MQWQR	
MQWQR_AREA	DS	CL(MQWQR_LENGTH)	

Odsyłacze pokrewne

[Pola w tabeli MQWQR -Struktura rekordu kolejki obciążenia klastra](#)

Opis pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra

W poniższej tabeli podsumowano pola w strukturze rekordu obciążenia klastra MQWCR .

<i>Tabela 202. Pola w MQWCR.</i>		
W tej tabeli znajdują się trzy kolumny. Pierwsza kolumna zawiera nazwy pól, druga kolumna zawiera opis każdego pola, a trzecia kolumna zawiera odsyłacz do dalszych informacji o każdej zmiennej.		
Pole	Opis	Strona
<i>ClusterName</i>	Nazwa klastra	ClusterName
<i>ClusterRecOffset</i>	Przesunięcie następnego rekordu klastra (MQWCR)	ClusterRecPrzesunięcie
<i>ClusterFlags</i>	Flagi klastra	ClusterFlags

Struktura rekordu klastra obciążenia klastra zawiera informacje na temat klastra. Dla każdego klastra, do którego należy kolejka docelowa, istnieje jedna struktura rekordu klastra obciążenia klastra.

Struktura rekordu klastra obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Odsyłacze pokrewne

[MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia](#)

Program obsługi wyjścia obciążenia klastra jest wywoływany przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

[MQXCLWLN -Nawiguj rekordy obciążenia klastra](#)

Wywołanie funkcji MQXCLWLN jest używane do przechodzenia przez łańcuchy rekordów MQWDR, MQWQR i MQWCR przechowywanych w pamięci podręcznej klastra.

[MQWXP -Struktura parametru wyjścia obciążenia klastra](#)

W poniższej tabeli podsumowano pola w strukturze parametru wyjścia obciążenia klastra MQWXP .

[MQWDR-Struktura rekordu miejsca docelowego obciążenia klastra](#)

W poniższej tabeli przedstawiono podsumowanie pól w strukturze rekordu miejsca docelowego obciążenia klastra MQWDR .

[MQWQR -Struktura rekordu kolejki obciążenia klastra](#)

W poniższej tabeli podsumowano pola w strukturze rekordu kolejki obciążenia klastra MQWQR .

Pola w strukturze rekordu klastra obciążenia klastra MQWCR .

Opis pól w strukturze rekordu klastra obciążenia klastra MQWCR .

ClusterName (MQCHAR48) -wejście

Nazwa klastra, do którego należy instancja kolejki docelowej, do której należy struktura MQWCR .
Instancja kolejki docelowej jest opisana za pomocą struktury MQWDR .

- Długość parametru ClusterName to MQ_CLUSTER_NAME_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście

Przesunięcie logiczne następnej struktury MQWCR .

- Jeśli nie ma więcej struktur MQWCR , wartość ClusterRecOffset wynosi zero.
- Przesunięcie jest mierzone w bajtach od początku struktury MQWCR .

ClusterFlags (MQLONG) -wejście

Opcje klastra wskazują właściwości menedżera kolejek identyfikowanego przez strukturę MQWCR .
Zdefiniowane są następujące opcje:

MQQMF_REPOSITORY_Q_MGR

Miejsce docelowe jest pełnym menedżerem kolejek repozytorium.

MQQMF_CLUSSDR_USER_DEFINED

Kanał wysyłający klastry został zdefiniowany ręcznie.

MQQMF_CLUSSDR_AUTO_DEFINED

Kanał wysyłający klastry został zdefiniowany automatycznie.

MQQMF_AVAILABLE

Docelowy menedżer kolejek jest dostępny do odbierania komunikatów.

Inne wartości

Inne flagi w tym polu mogą być ustawione przez menedżer kolejek do celów wewnętrznych.

Odsyłacze pokrewne

Wartości początkowe i deklaracje języka dla produktu MQWCR

Wartości początkowe i deklaracje języka C i High Level Assembler dla struktury rekordu klastra obciążenia klastra MQWCR .

Wartości początkowe i deklaracje języka dla produktu MQWCR

Wartości początkowe i deklaracje języka C i High Level Assembler dla struktury rekordu klastra obciążenia klastra MQWCR .

Tabela 203. Początkowe wartości pól w MQWCR		
Nazwa pola	Nazwa stałej	Wartość stałej
ClusterName	Brak	" "
ClusterRecOffset	Brak	0
ClusterFlags	Brak	0

Deklaracja C

```
typedef struct tagMQWCR {  
    MQCHAR48  ClusterName;      /* Cluster name */  
    MQLONG     ClusterRecOffset; /* Offset of next cluster record */  
    MQLONG     ClusterFlags;    /* Cluster flags */  
};
```

High Level Assembler

MQWCR

DSECT

MQWCR_CLUSTERNAME	DS	CL48	Cluster name
MQWCR_CLUSTERRECOFFSET	DS	F	Offset of next cluster
*			record
MQWCR_CLUSTERFLAGS	DS	F	Cluster flags
MQWCR_LENGTH	EQU	*-MQWCR	Length of structure
	ORG	MQWCR	
MQWCR_AREA	DS	CL(MQWCR_LENGTH)	

Odsyłacze pokrewne

Pola w strukturze rekordu klastra obciążenia klastra MQWCR .

Opis pól w strukturze rekordu klastra obciążenia klastra MQWCR .

Odwołanie do wyjścia funkcji API

Ta sekcja zawiera informacje uzupełniające, które są głównie interesujące dla programisty, który zapisuje wyjścia funkcji API.

Ogólne uwagi dotyczące użycia

uwagi:

1. Wszystkie funkcje wyjścia mogą wywoływać wywołanie MQXEP. To wywołanie jest zaprojektowane specjalnie do użytku z funkcji wyjścia funkcji API.
2. Funkcja MQ_INIT_EXIT nie może wywołać żadnych wywołań MQ innych niż wywołania MQXEP.
3. Nie można wywołać wywołania MQDISC dla bieżącego połączenia.
4. Jeśli funkcja wyjścia wysyła wywołanie MQCONN lub wywołanie MQCONNX z opcją MQCNO_HANDLE_SHARE_NONE, wywołanie kończy się z kodem przyczyny MQRC_ALREADY_CONNECTED, a zwracany uchwyt jest taki sam, jak parametr przekazany do wyjścia jako parametr.
5. Ogólnie, gdy funkcja wyjścia funkcji API wydaje wywołanie MQI, wyjścia funkcji API nie są wywoływane rekurencyjnie. Jeśli jednak funkcja wyjścia wydaje wywołanie MQCONNX z opcjami MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, wywołanie zwraca nowy współużytkowany uchwyt. Zapewnia to pakiet obsługi wyjścia z własnym uchwytem połączenia, a tym samym jednostkową pracę, która jest niezależna od jednostki pracy aplikacji. Pakiet obsługi wyjścia może używać tego uchwytu do umieszczania i pobierania komunikatów w obrębie własnej jednostki pracy, a także do zatwierdzania lub tworzenia kopii zapasowych tej jednostki pracy; wszystko to może być wykonane bez wpływu na jednostkę pracy aplikacji.

Ponieważ funkcja obsługi wyjścia korzysta z uchwytu połączenia innego niż uchwyt używany przez aplikację, wywołania funkcji MQ wywoływanych przez funkcję wyjścia powodują wywołanie odpowiednich funkcji wyjścia funkcji API. Dlatego funkcje wyjścia mogą być wywoływane rekurencyjnie. Należy zauważyć, że zarówno pole *ExitUserArea* w MQAXP, jak i obszar łańcucha wyjścia mają zasięg uchwytu połączenia. Dlatego funkcja wyjścia nie może użyć tych obszarów do zasygnalizacji innej instancji wywoływanej rekurencyjnie, że jest ona już aktywna.

6. Funkcje obsługi wyjścia mogą również umieszczać i dostawać komunikaty w obrębie jednostki pracy aplikacji. Gdy aplikacja zatwierdza lub wycofuje jednostkę pracy, wszystkie komunikaty w jednostce pracy są zatwierdzane lub wycofane razem, niezależnie od tego, kto umieścił je w jednostce pracy (funkcja aplikacji lub wyjścia). Jednak wyjście może spowodować, że aplikacja przekroczy limity systemowe szybciej, niż byłoby to inaczej (na przykład, przekraczając maksymalną liczbę niezatwierdzonych komunikatów w jednostce pracy).

Gdy funkcja wyjścia korzysta z jednostki pracy aplikacji w ten sposób, funkcja wyjścia powinna zwykle unikać wywoływania wywołania MQCMIT, ponieważ ta funkcja zatwierdza jednostkę pracy aplikacji i może zaburzać poprawne działanie aplikacji. Jednak czasami funkcja obsługi wyjścia może wymagać wywołania wywołania MQBACK, jeśli funkcja wyjścia napotka poważny błąd uniemożliwiający wykonanie jednostki pracy (na przykład błąd podczas umieszczania komunikatu jako część jednostki pracy aplikacji). Gdy wywoływane jest wywołanie MQBACK, należy zadbać o to, aby nie zostały zmienione granice pracy aplikacji. W takiej sytuacji funkcja wyjścia musi ustawić odpowiednie wartości w celu zapewnienia, że kod zakończenia MQCC_WARNING i kod przyczyny

MQRC_BACKED_OUT są zwracane do aplikacji, tak aby aplikacja mogła wykryć fakt, że jednostka pracy została wycofana.

Jeśli funkcja wyjścia korzysta z uchwytu połączenia aplikacji w celu wywołania wywołań MQ, te wywołania nie powodują kolejnych wywołań funkcji wyjścia funkcji API.

7. Jeśli funkcja wyjścia MQXR_BEFORE zakończy działanie w sposób nieprawidłowy, menedżer kolejek może być w stanie naprawić błąd z powodu niepowodzenia. Jeśli to możliwe, menedżer kolejek kontynuuje przetwarzanie tak, jakby funkcja obsługi wyjścia zwróciła błąd MQXCC_FAILED. Jeśli menedżer kolejek nie może odtworzyć danych, aplikacja zostanie zakończona.
8. Jeśli funkcja wyjścia MQXR_AFTER kończy działanie w sposób nieprawidłowy, menedżer kolejek może być w stanie naprawić błąd w wyniku niepowodzenia. Jeśli to możliwe, menedżer kolejek kontynuuje przetwarzanie tak, jakby funkcja obsługi wyjścia zwróciła błąd MQXCC_FAILED. Jeśli menedżer kolejek nie może odtworzyć danych, aplikacja zostanie zakończona. Należy pamiętać, że w tym ostatnim przypadku komunikaty pobierane poza jednostką pracy są tracone (jest to taka sama sytuacja, jak aplikacja, w przypadku której nie powiodła się natychmiast po usunięciu komunikatu z kolejki).
9. Proces MCA wykonuje zatwierdzanie dwufazowe.

Jeśli wyjście interfejsu API przechwytuje komunikat MQCMIT z przygotowanego procesu MCA i podejmie próbę wykonania działania w jednostce pracy, działanie zakończy się niepowodzeniem z kodem przyczyny MQRC_UOW_NOT_AVAILABLE.
10. W przypadku środowiska z wieloma instalkami jedynym sposobem na wyjście z produktu Websphere MQ Wersja 7.0 i Wersja 7.1 jest zapisanie wyjścia w sposób, który łączy się z produktem Wersja 7.0 z mqm.Lib, a w przypadku wyjść innych niż podstawowy lub rezlokizowany, aby upewnić się, że aplikacja znajdzie poprawny plik mqm.Lib dla instalacji, z którą menedżer kolejek jest obecnie powiązany, przed uruchomieniem aplikacji. (Na przykład przed uruchomieniem aplikacji należy uruchomić komendę **setmqenv -m QM**, nawet jeśli właścicielem menedżera kolejek jest instalacja produktu Wersja 7.0).
11. Jeśli dostępnych jest wiele instalacji produktu IBM MQ, należy użyć wyjść napisanych dla wcześniejszej wersji produktu IBM MQ, ponieważ nowe funkcje dodane w późniejszej wersji mogą nie działać z wcześniejszymi wersjami. Więcej informacji na temat zmian w wydaniach zawiera sekcja [Co się zmieniło w produkcie IBM MQ 8.0](#).

Struktura parametru wyjścia funkcji API produktu IBM MQ (MQAXP)

Struktura MQAXP, zewnętrzny blok sterujący, jest używany jako parametr wejściowy lub wyjściowy do wyjścia funkcji API. Ten temat zawiera również informacje na temat sposobu, w jaki menedżery kolejek mają funkcje wyjścia procesu.

Produkt MQAXP ma następującą deklarację C:

```
typedef struct tagMQAXP {
    MQCHAR4   StrucId;           /* Structure identifier */
    MQLONG    Version;           /* Structure version number */
    MQLONG    ExitId;            /* Exit Identifier */
    MQLONG    ExitReason;        /* Exit invocation reason */
    MQLONG    ExitResponse;      /* Response code from exit */
    MQLONG    ExitResponse2;     /* Secondary response code from exit */
    MQLONG    Feedback;          /* Feedback code from exit */
    MQLONG    APICallerType;     /* MQSeries API caller type */
    MQBYTE16  ExitUserArea;      /* User area for use by exit */
    MQCHAR32  ExitData;          /* Exit data area */
    MQCHAR48  ExitInfoName;      /* Exit information name */
    MQBYTE48  ExitPDArea;        /* Problem determination area */
    MQCHAR48  QMgrName;          /* Name of local queue manager */
    PMQACH    ExitChainAreaPtr;  /* Inter exit communication area */
    MQHCONFIG Hconfig;           /* Configuration handle */
    MQLONG    Function;          /* Function Identifier */
    /* Ver:1 */
    MQHMSG    ExitMsgHandle      /* Exit message handle */
    /* Ver:2 */
};
```

Po wywołaniu funkcji w wyjściu funkcji API przekazywana jest następująca lista parametrów:

StrucId (MQCHAR4)-dane wejściowe

Identyfikator struktury parametru wyjścia, którego wartość jest następująca:

```
MQAXP_STRUC_ID.
```

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

Wersja (MQLONG)-dane wejściowe

Numer wersji struktury, o wartości:

MQAXP_VERSION_1

Struktura parametru wyjścia funkcji API w wersji 1.

MQAXP_VERSION_2

Struktura parametru wyjścia funkcji API w wersji 2.

MQAXP_CURRENT_VERSION

Bieżący numer wersji dla struktury parametru wyjścia funkcji API.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

ExitId (MQLONG)-dane wejściowe

Identyfikator wyjścia ustawiony przy wpisach do procedury wyjścia, wskazujący typ wyjścia:

MQXT_API_EXIT

Wyjście funkcji API.

ExitReason (MQLONG)-dane wejściowe

Przyczyna wywołania wyjścia, ustawiona dla każdej z funkcji wyjścia:

MQXR_CONNECTION

Wyjście jest wywoływane w celu zainicjowania przed wywołaniem MQCONN lub MQCONNX lub do samego zakończenia po wywołaniu wywołania MQDISC.

MQXR_PRZED

Wyjście jest wywoływane przed wykonaniem wywołania funkcji API lub przed przekształceniem danych w MQGET.

MQXR_AFTER

Wyjście jest wywoływane po wywołaniu funkcji API.

ExitResponse (MQLONG)-dane wyjściowe

Odpowiedź z wyjścia, zainicjowana przy wpisie do każdej funkcji wyjścia, w celu:

MQXCC_OK

Kontynuuj normalnie.

To pole musi być ustawione przez funkcję wyjścia, aby komunikował się z menedżerem kolejek w wyniku wykonania funkcji wyjścia. Wartość musi być jedną z następujących wartości:

MQXCC_OK

Funkcja obsługi wyjścia została zakończona pomyślnie. Kontynuuj normalnie.

Ta wartość może być ustawiona przez wszystkie funkcje wyjścia MQXR_ *. ExitResponse2 służy do decydowania, czy w późniejszym czasie w łańcuchu mają być wywoływane funkcje wyjścia.

Niepowodzenie MQXCC_FAILED

Funkcja wyjścia nie powiodła się z powodu błędu.

Ta wartość może być ustawiona przez wszystkie funkcje wyjścia MQXR_ *. Menedżer kolejek ustawia wartość CompCode na wartość MQCC_FAILED, a przyczyna:

- MQRC_API_EXIT_INIT_ERROR, jeśli funkcja ma wartość MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR, jeśli funkcja ma wartość MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR dla wszystkich pozostałych funkcji wyjścia

Zestaw wartości może zostać zmieniony przez funkcję wyjścia w późniejszym czasie w łańcuchu.

ExitResponse2 jest ignorowane; menedżer kolejek kontynuuje przetwarzanie, ponieważ został zwrócony łańcuch MQXR2_SUPPRESS_CHAIN .

MQXCC_SUPPRESS_FUNCTION

Pomijaj funkcję API IBM MQ .

Tę wartość można ustawić tylko za pomocą funkcji wyjścia MQXR_BEFORE. Pomija wywołanie interfejsu API. Jeśli jest zwracana przez program MQ_DATA_CONV_ON_GET_EXIT, konwersja danych jest pomijana. Menedżer kolejek ustawia wartość CompCode na MQCC_FAILED, a wartość MQRC_SUPPRESSED_BY_EXIT, ale zestaw wartości może zostać zmieniony przez funkcję wyjścia w późniejszym czasie w łańcuchu. Pozostałe parametry dla połączenia pozostają jak wyjście z nich. ExitResponse2 służy do decydowania, czy w późniejszym czasie w łańcuchu mają być wywoływane funkcje wyjścia.

Jeśli ta wartość jest ustawiona za pomocą funkcji wyjścia MQXR_AFTER lub MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie, ponieważ komenda MQXCC_FAILED nie została zwrócona.

MQXCC_SKIP_FUNCTION,

Pomiń funkcję API IBM MQ .

Tę wartość można ustawić tylko za pomocą funkcji wyjścia MQXR_BEFORE. Pomija wywołanie interfejsu API. Jeśli jest zwracana przez program MQ_DATA_CONV_ON_GET_EXIT, konwersja danych jest pomijana. Funkcja wyjścia musi ustawić parametr CompCode i powód, aby wartości były zwracane do aplikacji, ale zestaw wartości może zostać zmieniony przez funkcję wyjścia w późniejszym czasie w łańcuchu. Pozostałe parametry dla połączenia pozostają jak wyjście z nich. ExitResponse2 służy do decydowania, czy w późniejszym czasie w łańcuchu mają być wywoływane funkcje wyjścia.

Jeśli ta wartość jest ustawiona za pomocą funkcji wyjścia MQXR_AFTER lub MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie, ponieważ komenda MQXCC_FAILED nie została zwrócona.

MQXCC_SUPPRESS_EXIT

Pomijaj wszystkie funkcje wyjścia należące do zestawu wyjść.

Ta wartość może być ustawiona tylko przez funkcje wyjścia MQXR_BEFORE i MQXR_AFTER. Pomija on *wszystkie* kolejne wywołania funkcji wyjścia należących do tego zestawu wyjść dla tego połączenia logicznego. Pomijanie jest kontynuowane do momentu, gdy wystąpi żądanie rozłączenia logicznego, gdy funkcja MQ_TERM_EXIT jest wywoływana z wartością ExitReason MQXR_CONNECTION.

Funkcja wyjścia musi ustawić parametr CompCode i powód, aby wartości były zwracane do aplikacji, ale zestaw wartości może zostać zmieniony przez funkcję wyjścia w późniejszym czasie w łańcuchu. Pozostałe parametry dla połączenia pozostają jak wyjście z nich. ExitResponse2 jest ignorowane.

Jeśli ta wartość jest ustawiona za pomocą funkcji wyjścia MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie, ponieważ komenda MQXCC_FAILED nie została zwrócona.

Więcej informacji na temat interakcji między ExitResponse i ExitResponse2 oraz jego wpływu na przetwarzanie wyjścia zawiera sekcja [“Sposób zarządzania funkcjami wyjścia przez menedżery kolejek”](#) na stronie 1658.

ExitResponse2 (MQLONG)-dane wyjściowe

Jest to dodatkowy kod odpowiedzi wyjścia, który kwalifikuje podstawowy kod odpowiedzi wyjścia dla funkcji wyjścia MQXR_BEFORE. Jest on inicjowany do:

```
MQXR2_DEFAULT_CONTINUATION
```

przy wpisie do funkcji wyjścia wywołania funkcji API IBM MQ . Można go następnie ustawić na jedną z wartości:

MQXR2_DEFAULT_CONTINUATION

Określa, czy kontynuować przy następnym wyjściu w łańcuchu, w zależności od wartości parametru ExitResponse.

Jeśli ExitResponse to MQXCC_SUPPRESS_FUNCTION lub MQXCC_SKIP_FUNCTION, pomijanie funkcji wyjścia w późniejszym czasie w łańcuchu MQXR_BEFORE oraz zgodne funkcje wyjścia w łańcuchu MQXR_AFTER. Wywołaj funkcje wyjścia w łańcuchu MQXR_AFTER, które są zgodne z funkcjami wyjścia wcześniej w łańcuchu MQXR_BEFORE.

W przeciwnym razie wywołaj następne wyjście w łańcuchu.

MQXR2_SUPPRESS_CHAIN

Pomijaj łańcuch.

Obejście funkcji wyjścia w późniejszym czasie w łańcuchu MQXR_BEFORE i zgodnych funkcji wyjścia w łańcuchu MQXR_AFTER dla tego wywołania wywołania API. Wywołaj funkcje wyjścia w łańcuchu MQXR_AFTER, które są zgodne z funkcjami wyjścia wcześniej w łańcuchu MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN

Przejdź do następnego wyjścia w łańcuchu.

Więcej informacji na temat interakcji między ExitResponse i ExitResponse2 oraz jego wpływu na przetwarzanie wyjścia zawiera sekcja [“Sposób zarządzania funkcjami wyjścia przez menedżery kolejek”](#) na stronie 1658.

Feedback (MQLONG)-input/output

Komunikowanie kodów sprzężenia zwrotnego między wywołaniami funkcji wyjścia. Ta opcja jest inicjowana:

```
MQFB_NONE (0)
```

przed wywołaniem pierwszej funkcji pierwszego wyjścia w łańcuchu.

Wyjścia można ustawić w tym polu na dowolną wartość, w tym wszystkie poprawne wartości MQFB_* lub MQRC_*. Wyjścia mogą również ustawić to pole na wartość informacji zwrotnej zdefiniowanej przez użytkownika z zakresu MQFB_APPL_FIRST na MQFB_APPL_LAST.

APICallerType (MQLONG)-dane wejściowe

Typ programu wywołującego interfejsu API, który wskazuje, czy program wywołujący interfejs API produktu IBM MQ jest zewnętrzny, czy wewnętrzny w menedżerze kolejek: MQXACT_EXTERNAL lub MQXACT_INTERNAL.

ExitUserArea (MQBYTE16)-input/output

Obszar użytkownika, dostępny dla wszystkich wyjść powiązanych z konkretnym obiektem ExitInfo. Jest on inicjowany do wywołania MQXUA_NONE (zera binarne dla długości obszaru ExitUser) przed wywołaniem pierwszej funkcji wyjścia (MQ_INIT_EXIT) dla hconn. Od tego czasu wszystkie zmiany wprowadzone w tym polu przez funkcję wyjścia są zachowywane w różnych wywołaniach funkcji tego samego wyjścia.

To pole jest wyrównane do wielokrotności 4 MQLONGs.

Wyjścia mogą również zakotwiczenie dowolnej pamięci masowej, którą przydzieli z tego obszaru.

Dla każdego hconn każde wyjście w łańcuchu wyjść ma inny obszar ExitUser. Obszar ExitUser nie może być współużytkowany przez wyjścia w łańcuchu, a zawartość obszaru ExitUser dla jednego wyjścia nie jest dostępna dla innego wyjścia w łańcuchu.

W przypadku programów w języku C stała MQXUA_NONE_ARRAY jest również zdefiniowana z tą samą wartością, co MQXUA_NONE, ale jako tablica znaków zamiast łańcucha.

Długość tego pola jest podana przez wartość MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32)-dane wejściowe

Wydź z danych, ustaw dane wejściowe dla każdej funkcji wyjścia na 32 znaki danych specyficznych dla wyjścia, które są podane w wyjściu. Jeśli w wyjściu nie zostanie zdefiniowana żadna wartość, to pole będzie puste.

Długość tego pola jest podana przez wartość MQ_EXIT_DATA_LENGTH.

Nazwa ExitInfo(MQCHAR48)-dane wejściowe

Nazwa informacji o wyjściu, ustawiana na danych wejściowych dla każdej funkcji wyjścia na wartość ApiExit_name określona w definicjach wyjścia w sekcjach.

ExitPDArea (MQBYTE48)-wejście/wyjście

Obszar określania problemu, zainicjowany na MQXPDA_NONE (binarne zera dla długości pola) dla każdego wywołania funkcji wyjścia.

W przypadku programów w języku C stała MQXPDA_NONE_ARRAY jest również zdefiniowana z tą samą wartością, co MQXPDA_NONE, ale jako tablica znaków zamiast łańcucha.

Procedura obsługi wyjścia zawsze zapisuje ten obszar w pliku śledzenia produktu IBM MQ na końcu wyjścia, nawet jeśli funkcja jest pomyślnie wykonana.

Długość tego pola jest podana przez wartość MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek, z którym połączona jest aplikacja, która wywołała wyjście w wyniku przetwarzania wywołania funkcji API produktu IBM MQ .

Jeśli nazwa menedżera kolejek dostarczonego w wywołaniach MQCONN lub MQCONNx jest pusta, to pole jest nadal ustawione na nazwę menedżera kolejek, z którym połączona jest aplikacja, niezależnie od tego, czy aplikacja jest serwerem, czy klientem.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

Długość tego pola jest podana przez wartość MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH)-wejście/wyjście

Jest on używany do przekazywania danych między wywołaniami różnych wyjść w łańcuchu. Jest on ustawiony na pusty wskaźnik przed wywołaniem pierwszej funkcji (MQ_INIT_EXIT z ExitReason MQXR_CONNECTION) pierwszego wyjścia w łańcuchu wyjść. Wartość zwrócona przez wyjście w jednym wywołaniu jest przekazywana do następnego wywołania.

Więcej informacji na temat korzystania z obszaru łańcucha wyjścia można znaleźć w sekcji [“Obszar łańcucha wyjścia i nagłówek obszaru łańcucha wyjścia \(MQACH\)”](#) na stronie 1661 .

Hconfig (MQHCONFIG)-dane wejściowe

Uchwyty konfiguracji reprezentujący zestaw inicjowanych funkcji. Ta wartość jest generowana przez menedżer kolejek w funkcji MQ_INIT_EXIT, a następnie jest przekazywana do funkcji wyjścia funkcji API. Jest ona ustawiana przy wpisach do każdej funkcji wyjścia.

Hconfig można użyć jako wskaźnika do struktury MQIEP w celu wykonania wywołań MQI i DCI. Przed użyciem parametru Hconfig jako wskaźnika do struktury MQIEP należy sprawdzić, czy pierwsze 4 bajty Hconfig są zgodne ze strukturą StrucId struktury MQIEP.

Funkcja (MQLONG)-dane wejściowe

Identyfikator funkcji, poprawne wartości, dla których są stałe MQXF_ * opisane w [“Stałe zewnętrzne”](#) na stronie 1663.

Procedura obsługi wyjścia ustawia to pole na poprawną wartość przy wpisach do każdej funkcji wyjścia, w zależności od wywołania funkcji API IBM MQ , które spowodowało wywołanie wyjścia.

Uchwyty ExitMsg(MQHMSG)-input/output

Jeśli funkcja to MQXF_GET i ExitReason to MQXR_AFTER, w tym polu zwracany jest poprawny uchwyt komunikatu, co pozwala na dostęp do pól deskryptora komunikatu przez interfejs API oraz wszelkie inne właściwości zgodne z łańcuchem ExitProperties określonym w strukturze MQXEPO podczas rejestrowania wyjścia funkcji API.

Wszystkie właściwości deskryptora niezwiązane z komunikatami, które są zwracane w uchwycie ExitMsg, nie będą dostępne z elementu MsgHandle w strukturze MQGMO, jeśli został określony, lub w danych komunikatu.

Jeśli funkcja to MQXF_GET i ExitReason ma wartość MQXR_BEFORE, jeśli program obsługi wyjścia ustawia to pole na wartość MQHM_NONE, wówczas pominięte zostanie wypełnianie właściwości uchwytu ExitMsg.

To pole nie jest ustawione, jeśli wersja jest mniejsza niż MQAXP_VERSION_2.

Sposób zarządzania funkcjami wyjścia przez menedżery kolejek

Przetwarzanie wykonywane przez menedżer kolejek po powrocie z funkcji wyjścia jest zależne od wartości ExitResponse i ExitResponse2.

Tabela 204 na stronie 1658 podsumowuje możliwe kombinacje i ich efekty dla funkcji wyjścia MQXR_BEFORE, pokazując:

- Kto ustawia parametry CompCode i Reason w wywołaniu API
- Określa, czy pozostałe funkcje wyjścia w łańcuchu MQXR_BEFORE i zgodne funkcje wyjścia w łańcuchu MQXR_AFTER są wywoływane.
- Określa, czy wywołanie API zostało wywołane

Dla funkcji wyjścia MQXR_AFTER:

- Parametr CompCode i Przyczyna są ustawione w taki sam sposób, jak MQXR_BEFORE
- ExitResponse2 jest ignorowane (pozostałe funkcje wyjścia w łańcuchu MQXR_AFTER są zawsze wywoływane)
- MQXCC_SUPPRESS_FUNCTION i MQXCC_SKIP_FUNCTION nie są poprawne

Dla funkcji wyjścia MQXR_CONNECTION:

- Parametr CompCode i Przyczyna są ustawione w taki sam sposób, jak MQXR_BEFORE
- ExitResponse2 jest ignorowane
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT są niepoprawne

We wszystkich przypadkach, w których wyjście lub menedżer kolejek ustawia CompCode i Reason, zestaw wartości może zostać zmieniony przez wyjście wywołane później lub przez wywołanie API (jeśli wywołanie API zostało później wywołane).

Tabela 204. Przetwarzanie wyjścia MQXR_BEFORE			
Wartość parametru ExitResponse	CompCode i przyczyna ustawiona przez	Wartość łańcucha ExitResponse2 (domyślna kontynuacja)	Wartość funkcji API ExitResponse2 (kontynuacja domyślna)
MQXCC_OK	exit	Y	Y
MQXCC_SUPPRESS_EXIT	exit	Y	Y
MQXCC_SUPPRESS_FUNCTION	menedżer kolejek	N	N
MQXCC_SKIP, FUNKCJA	exit	N	N
Niepowodzenie MQXCC_FAILED	menedżer kolejek	N	N

Sposób przetwarzania przez klientów funkcji wyjścia

W ogólnym przypadku funkcje obsługi wyjścia procesu klienta są takie same, jak aplikacje serwera, a atrybut QMgrName w tej strukturze ma zastosowanie, czy funkcja znajduje się na serwerze, czy na kliencie.

Jednak klient nie ma pojęcia w pliku *mqs.ini* , dlatego nie mają zastosowania sekcje *ApiExitCommon* i *APIExitTemplate* . Zastosowanie ma tylko sekcja *ApiExitLocal* , a ta sekcja jest skonfigurowana w pliku *mqclient.ini* .

Struktura kontekstu wyjścia funkcji API produktu IBM MQ (MQAXC)

Struktura MQAXC, zewnętrzny blok sterujący, jest używany jako parametr wejściowy do wyjścia funkcji API.

MQAXC ma następującą deklarację C:

```
typedef struct tagMQAXC {
    MQCHAR4    StrucId;                /* Structure identifier */
    MQLONG     Version;                /* Structure version number */
    MQLONG     Environment;            /* Environment */
    MQCHAR12    UserId;                /* UserId associated with appl */
    MQBYTE40    SecurityId;            /* Extension to UserId running appl */
    MQCHAR264    ConnectionName;        /* Connection name */
    MQLONG     LongMCAUserIdLength;    /* long MCA user identifier length */
    MQLONG     LongRemoteUserIdLength; /* long remote user identifier length */
    MQPTR       LongMCAUserIdPtr;      /* long MCA user identifier address */
    MQPTR       LongRemoteUserIdPtr;   /* long remote user identifier address */
    MQCHAR28    ApplName;              /* Application name */
    MQLONG     ApplType;               /* Application type */
    MQPID       ProcessId;              /* Process identifier */
    MQTID       ThreadId;              /* Thread identifier */

    /* Ver:1 */
    MQCHAR     ChannelName[20]         /* Channel Name */
    MQBYTE4    Reserved1;              /* Reserved */
    PMQCD      pChannelDefinition;     /* Channel Definition pointer */
};
```

Parametry do wywołania MQAXC to:

StrucId (MQCHAR4)-dane wejściowe

Identyfikator struktury kontekstu wyjścia, którego wartość jest MQAXC_STRUC_ID. W przypadku programów w języku C stała jest również zdefiniowana stała MQAXC_STRUC_ID_ARRAY, o tej samej wartości co identyfikator MQAXC_STRUC_ID, ale jako tablica znaków zamiast łańcucha.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

Wersja (MQLONG)-dane wejściowe

Numer wersji struktury, o wartości:

MQAXC_VERSION_2

Numer wersji struktury kontekstu wyjścia.

MQAXC_CURRENT_VERSION

Bieżący numer wersji dla struktury kontekstu wyjścia.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

Środowisko (MQLONG)-dane wejściowe

Środowisko, z którego wywołano wywołanie funkcji API IBM MQ , które spowodowało, że funkcja wyjścia jest sterowana. Poprawne wartości dla tego pola to:

MQXE_INNY

Ta wartość jest spójna z wywołaniami wyjścia funkcji API, jeśli wyjście jest wywoływane z poziomu aplikacji serwera. Oznacza to, że wyjście interfejsu API działa bez zmian na kliencie i nie widzi niczego innego.

Jeśli wyjście naprawdę wymaga określenia, czy jest on uruchomiony na kliencie, wyjście może to zrobić, przeglądając pola *ChannelName* i *ChannelDefinition* .

MQXE_MCA

Agent kanału komunikatów

MQXE_MCA_SVRCONN

Agent kanału komunikatów działający w imieniu klienta

MQXE_COMMAND_SERVER

Serwer komend

MQXE_MQSC

Interpreter komend runmqsc

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

UserId (MQCHAR12)-dane wejściowe

Identyfikator użytkownika powiązany z aplikacją. W szczególności w przypadku połączeń klienckich pole to zawiera identyfikator użytkownika adoptowanych użytkowników w przeciwieństwie do ID użytkownika, pod którym kod kanału jest uruchomiony. Jeśli z klienta przepływa pusty identyfikator użytkownika, nie zostanie dokonana żadna zmiana dla ID użytkownika, który już jest używany. Oznacza to, że nie jest adoptowane żadne nowe ID użytkownika.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia. Długość tego pola jest podana przez wartość MQ_USER_ID_LENGTH.

W przypadku klienta jest to identyfikator użytkownika wysłany z klienta na serwer. Należy zauważyć, że może to nie być efektywny identyfikator użytkownika, dla którego klient jest uruchamiany w menedżerze kolejek, ponieważ może to być konfiguracja MCAUser lub CHLAUTH, która zmienia ID użytkownika.

SecurityId (MQBYTE40)-dane wejściowe

Rozszerzenie ID użytkownika uruchamiające aplikację. Jej długość jest nadawana przez wartość MQ_SECURITY_ID_LENGTH.

W przypadku klienta jest to identyfikator użytkownika wysłany z klienta na serwer. Należy zauważyć, że może to nie być efektywny identyfikator użytkownika, dla którego klient jest uruchamiany w menedżerze kolejek, ponieważ może to być konfiguracja MCAUser lub CHLAUTH, która zmienia ID użytkownika.

ConnectionName (MQCHAR264)-dane wejściowe

Pole nazwy połączenia, ustawione na adres klienta. Na przykład w przypadku protokołu TCP/IP adres IP klienta jest taki sam.

Długość tego pola jest podana przez wartość MQ_CONN_NAME_LENGTH.

W przypadku klienta jest to adres partnera menedżera kolejek.

LongMCAUserIdLength (MQLONG)-dane wejściowe

Długość identyfikatora użytkownika długiego MCA.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawione na długość identyfikatora użytkownika długiego MCA (lub wartość 0, jeśli taki identyfikator nie istnieje).

W przypadku klienta jest to długi identyfikator użytkownika klienta.

LongRemoteUserIdLength (MQLONG)-wejście

Długość długiego identyfikatora zdalnego użytkownika.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawione na długość długiego identyfikatora zdalnego użytkownika. W przeciwnym razie to pole zostanie ustawione na zero.

W przypadku klienta ustaw w tym polu wartość zero.

LongMCAUserIdPtr (MQPTR)-dane wejściowe

Adres identyfikatora użytkownika długiego MCA.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawione na adres identyfikatora użytkownika długiego MCA (lub do pustego wskaźnika, jeśli taki identyfikator nie istnieje).

W przypadku klienta jest to długi identyfikator użytkownika klienta.

LongRemoteUserIdPtr (MQPTR)-wejście

Adres długiego zdalnego identyfikatora użytkownika.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawione na adres długiego zdalnego identyfikatora użytkownika (lub do wskaźnika pustego, jeśli taki identyfikator nie istnieje).

W przypadku klienta ustaw w tym polu wartość zero.

ApplName (MQCHAR28)-dane wejściowe

Nazwa aplikacji lub komponentu, który wywołał wywołanie funkcji API produktu IBM MQ .

Reguły generowania parametru ApplName są takie same, jak w przypadku generowania domyślnej nazwy dla MQPUT.

Wartość tego pola można znaleźć, wysyłając zapytanie do systemu operacyjnego o nazwę programu. Jego długość jest podana przez wartość MQ_APPL_NAME_LENGTH.

ApplType (MQLONG)-dane wejściowe

Typ aplikacji lub komponentu, który wywołał wywołanie funkcji API IBM MQ .

Wartość jest równa MQAT_DEFAULT dla platformy, na której jest kompilowana aplikacja, lub jest równa jednej z zdefiniowanych wartości MQAT_ *.

Procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

ProcessId (MQPID)-dane wejściowe

Identyfikator procesu systemu operacyjnego.

Tam, gdzie ma to zastosowanie, procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

ThreadId (MQTID)-dane wejściowe

Identyfikator wątku MQ . Jest to ten sam identyfikator używany w danych śledzenia MQ i zrzutach FFST , ale może być inny niż identyfikator wątku systemu operacyjnego.

Tam, gdzie ma to zastosowanie, procedura obsługi wyjścia ustawia to pole przy wpisach do każdej funkcji wyjścia.

ChannelName (MQCHAR)-dane wejściowe

Nazwa kanału, dopełniona odstępami (jeśli ma zastosowanie i jest znana).

Jeśli nie ma zastosowania, to pole jest ustawione na znaki NULL.

Reserved1 (MQBYTE4)-dane wejściowe

To pole jest zarezerwowane.

ChanneDefinition (PMQCD)-dane wejściowe

Wskaźnik do używanej definicji kanału, jeśli ma zastosowanie i jest znana.

Jeśli nie ma zastosowania, to pole jest ustawione na znaki NULL.

Należy zauważyć, że wskaźnik jest wypełniony tylko wtedy, gdy połączenie jest przetwarzane w imieniu kanału IBM MQ i ta definicja kanału została odczyta.

W szczególności definicja kanału nie jest podana na serwerze, gdy dla kanału zostanie wykonane pierwsze wywołanie MQCONN. Co więcej, jeśli wskaźnik jest wypełniony, struktura (i wszelkie podstruktury) wskazywana przez wskaźnik musi być traktowana jako tylko do odczytu; każda aktualizacja struktury doprowadziłaby do nieprzewidywalnych wyników i nie jest obsługiwana.

W przypadku klienta, pola inne niż te, które mają wartość określoną dla klienta, zawierają wartości odpowiednie dla aplikacji klienckiej.

Obszar łańcucha wyjścia i nagłówek obszaru łańcucha wyjścia (MQACH)

Jeśli jest to wymagane, funkcja wyjścia może uzyskać pamięć masową dla obszaru łańcucha wyjścia i ustawić parametr ExitChainAreaPtr w MQAXP, aby wskazywać na tę pamięć.

Wyjścia (te same lub różne funkcje wyjścia) mogą uzyskać wiele obszarów łańcucha wyjścia i połączyć je ze sobą. Obszary łańcucha wyjścia muszą zostać dodane lub usunięte tylko z tej listy podczas wywołania

z procedury obsługi wyjścia. Dzięki temu nie występują problemy związane z serializacją powodowane przez różne wątki, które powodują dodawanie lub usuwanie obszarów z listy w tym samym czasie.

Obszar łańcucha wyjścia musi zaczynać się od struktury nagłówka MQACH, dla której deklaracja C jest następująca:

```
typedef struct tagMQACH {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQLONG     StrucLength;       /* Length of the MQACH structure */
    MQLONG     ChainAreaLength;   /* Exit chain area length */
    MQCHAR48   ExitInfoName;     /* Exit information name */
    PMQACH     NextChainAreaPtr; /* Pointer to next exit chain area */
};
```

Pola w nagłówku obszaru łańcucha wyjścia są następujące:

StrucId (MQCHAR4)-dane wejściowe

Identyfikator struktury obszaru łańcucha wyjścia, z wartością początkową zdefiniowaną przez MQACH_DEFAULT, MQACH_STRUC_ID.

W przypadku programów w języku C jest również zdefiniowana stała MQACH_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQACH_STRUC_ID, ale jako tablica znaków zamiast łańcucha.

Wersja (MQLONG)-dane wejściowe

Numer wersji struktury w następujący sposób:

MQACH_VERSION_1

Numer wersji struktury parametru wyjścia.

MQACH_CURRENT_VERSION

Bieżący numer wersji dla struktury kontekstu wyjścia.

Początkowa wartość tego pola, zdefiniowana przez MQACH_DEFAULT, to MQACH_CURRENT_VERSION.

Uwaga: Jeśli wprowadzisz nową wersję tej struktury, układ istniejącej części się nie zmieni. Funkcje wyjścia muszą sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji zawierającej pola, których musi używać funkcja obsługi wyjścia.

StrucLength (MQLONG)-dane wejściowe

Długość struktury MQACH. Wyjścia mogą używać tego pola do określenia początku danych wyjściowych, ustawiając go na długość struktury utworzonej przez wyjście.

Początkowa wartość tego pola, zdefiniowana przez MQACH_DEFAULT, to MQACH_CURRENT_LENGTH.

ChainAreaLength (MQLONG)-input

Długość obszaru łańcucha wyjścia, ustawiona na całkowitą długość bieżącego obszaru łańcucha wyjścia, w tym nagłówki MQACH.

Początkowa wartość tego pola, zdefiniowana przez MQACH_DEFAULT, wynosi zero.

Nazwa ExitInfo(MQCHAR48)-dane wejściowe

Nazwa informacji o wyjściu.

Gdy wyjście tworzy strukturę MQACH, musi inicjować to pole za pomocą własnej nazwy ExitInfo, tak aby później tę strukturę MQACH można było znaleźć w innej instancji tego wyjścia lub przez współpracujące wyjście.

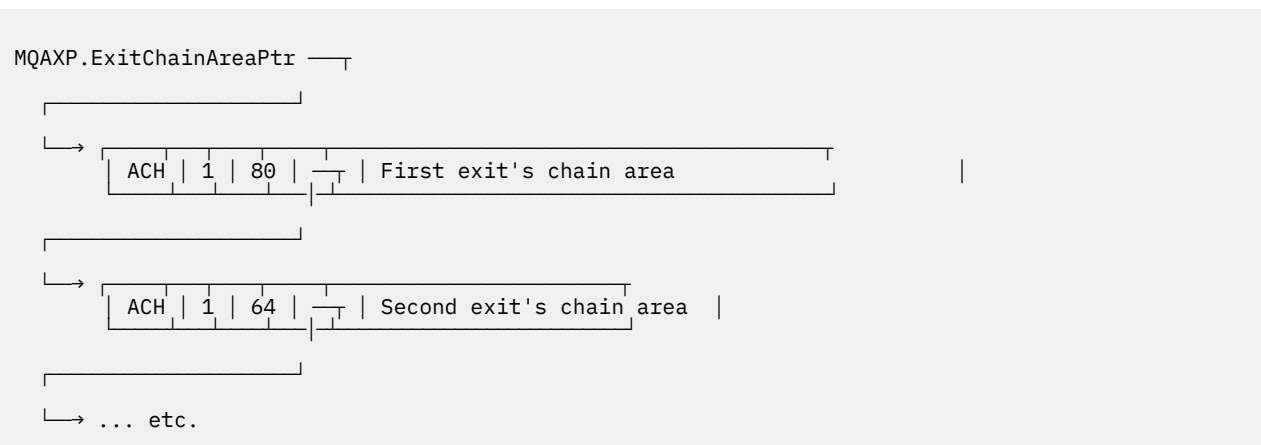
Początkowa wartość tego pola, zdefiniowana przez MQACH_DEFAULT, jest łańcuchem o zerowej długości ({}).

NextChainAreaPtr (PMQACH)-dane wejściowe

Wskaźnik do następnego obszaru łańcucha wyjścia o wartości początkowej zdefiniowanej przez parametr MQACH_DEFAULT, wskaźnik pusty (NULL).

Funkcje wyjścia muszą zwolnić pamięć dla wszystkich obszarów łańcucha wyjścia, które uzyskują, a także manipulować wskaźnikami łańcucha w celu usunięcia ich obszarów łańcucha wyjścia z listy.

Obszar łańcucha wyjścia może być skonstruowany w następujący sposób:



Stałe zewnętrzne

Ten temat zawiera informacje uzupełniające dotyczące stałych zewnętrznych dostępnych dla interfejsu API.

Dla wyjść funkcji API dostępne są następujące stałe zewnętrzne:

MQXF_* (identyfikatory funkcji wyjścia)

MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_CONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DISC	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_CLOSE	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
MQXF_GET	10	X'0000000A'
MQXF_DATA_CONV_ON_GET	11	X'0000000B'
MQXF_INQ	12	X'0000000C'
MQXF_SET	13	X'0000000D'
MQXF_BEGIN	14	X'0000000E'
MQXF_CMIT	15	X'0000000F'
MQXF_BACK	16	X'00000010'
MQXF_STAT	18	X'00000012'
MQXF_CB	19	X'00000013'
MQXF_CTL	20	X'00000014'
MQXF_CALLBACK	21	X'00000015'
MQXF_SUB	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
MQXF_XACLOSE	24	X'00000018'
MQXF_XACOMMIT	25	X'00000019'
MQXF_XACOMplete	26	X'0000001A'
MQXF_XAEND	27	X'0000001B'
MQXF_XAFORGET	28	X'0000001C'
MQXF_XAOPEN	29	X'0000001D'
MQXF_XAPREPARE	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
MQXF_XASTART	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXR_* (przyczyny wyjścia)

MQXR_BEFORE	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'

MQXE_* (środowiska)

MQXE_OTHER	0	X'00000000'
MQXE_MCA	1	X'00000001'

MQXE_MCA_SVRCONN	2	X'00000002'
MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQ*_*(dodatkowe state)

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*_*(state puste)

MQXPDA_NONE	X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY	'\0','\0',...,'\0','\0'

MQXCC_*(kody zakończenia)

MQXCC_FAILED	-8

MQRC_*(kody przyczyny)

MQRC_API_EXIT_ERROR 2374 X'00000946'

Wywołanie funkcji wyjścia zwróciło niepoprawny kod odpowiedzi lub w jakiś sposób nie powiodło się, a menedżer kolejek nie może określić następnego działania do wykonania.

Sprawdź pola ExitResponse i ExitResponse2 w MQAXP, aby określić zły kod odpowiedzi, a następnie zmień wyjście, aby zwracał poprawny kod odpowiedzi.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'

Menedżer kolejek napotkał błąd podczas inicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'

Menedżer kolejek napotkał błąd podczas zamykania środowiska wykonawczego dla funkcji wyjścia funkcji API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'

Wartość pola ExitReason podana w wywołaniu funkcji rejestrowania punktu wejścia wyjścia (MQXEP) jest błędna.

Sprawdź wartość w polu ExitReason, aby określić i poprawić błędną wartość przyczyny wyjścia.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'

Wartość pola Zarezerwowane jest błędna.

Sprawdź wartość pola Zarezerwowane, aby określić i poprawić wartość zarezerwowaną.

Typedefs języka C

Ten temat zawiera informacje na temat typów typów powiązanych z wyjściami interfejsu API dostępnych w języku C.

Oto typy kodu języka C powiązane z wyjściami interfejsu API:

```
typedef PMQLONG MQPOINTER PPMQLONG;
typedef PMQBYTE MQPOINTER PPMQBYTE;
typedef PMQHOBJS MQPOINTER PPMQHOBJS;
typedef PMQOD MQPOINTER PPMQOD;
typedef PMQMD MQPOINTER PPMQMD;
typedef PMQPMO MQPOINTER PPMQPMO;
typedef PMQGM0 MQPOINTER PPMQGM0;
typedef PMQCNO MQPOINTER PPMQCNO;
typedef PMQBO MQPOINTER PPMQBO;

typedef MQAXP MQPOINTER PMQAXP;
typedef MQACH MQPOINTER PMQACH;
typedef MQAXC MQPOINTER PMQAXC;

typedef MQCHAR MQCHAR16[16];
typedef MQCHAR16 MQPOINTER PMQCHAR16;

typedef MQLONG MQPID;
typedef MQLONG MQTID;
```

Wywołanie procedury rejestracji punktu wejścia wyjścia (MQXEP)

Te informacje umożliwiają zapoznanie się z produktem MQXEP, MQXEP C wywołania języka i prototypem funkcji produktu MQXEP C.

Użyj wywołania MQXEP do:

1. Zarejestruj przed i po IBM MQ punkty wywołania wyjścia funkcji API, w których mają być wywoływane funkcje wyjścia
2. Określ punkty wejścia funkcji wyjścia
3. Wyrejestruj punkty wejścia funkcji wyjścia

Zwykle kod wywołania MQXEP jest zakodowany w funkcji wyjścia MQ_INIT_EXIT, ale można je określić w dowolnej późniejszej funkcji wyjścia.

Jeśli do zarejestrowania już zarejestrowanej funkcji wyjścia zostanie użyte wywołanie MQXEP, to drugie wywołanie MQXEP zakończy się pomyślnie, zastępując zarejestrowaną funkcję wyjścia.

W przypadku użycia wywołania MQXEP w celu zarejestrowania funkcji wyjścia o wartości NULL wywołanie MQXEP zakończy się pomyślnie, a funkcja obsługi wyjścia jest wyrejestrowana.

Jeśli wywołania MQXEP są używane do rejestrowania, wyrejestrowywania i ponownego rejestrowania określonej funkcji wyjścia w czasie życia żądania połączenia, to poprzednio zarejestrowana funkcja wyjścia jest reaktywowana. Wszystkie pamięci nadal przydzielone i powiązane z tą instancją funkcji wyjścia są dostępne do użycia przez funkcje wyjścia. (Ta pamięć jest zwykle zwalniana podczas wywoływania funkcji wyjścia kończenia).

Interfejs do wywołania MQXEP jest następujący:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

gdzie:

Hconfig (MQHCONFIG)-dane wejściowe

Uchwyt konfiguracji reprezentujący wyjście funkcji API, które zawiera zestaw inicjowanych funkcji. Wartość ta jest generowana przez menedżer kolejek bezpośrednio przed wywołaniem funkcji MQ_INIT_EXIT i jest przekazywana w MQAXP do każdej funkcji wyjścia funkcji API.

ExitReason (MQLONG)-dane wejściowe

Przyczyna, dla której rejestrowany jest punkt wejścia, z następujących powodów:

- Inicjowanie lub kończenie na poziomie połączenia (MQXR_CONNECTION)
- Przed wywołaniem funkcji API produktu IBM MQ (MQXR_BEFORE)
- Po wywołaniu funkcji API IBM MQ (MQXR_AFTER)

Funkcja (MQLONG)-dane wejściowe

Identyfikator funkcji, poprawne wartości, dla których są stałe MQXF_* (patrz [“Stałe zewnętrzne”](#) na stronie 1663).

EntryPoint (PMQFUNC)-dane wejściowe

Adres punktu wejścia dla funkcji wyjścia, która ma zostać zarejestrowana. Wartość NULL wskazuje, że funkcja wyjścia nie została podana lub że wcześniejsza rejestracja funkcji wyjścia jest wyrejestrowana.

ExitOpts(MQXEPO)

Wyjścia funkcji API mogą określać opcje, które sterują rejestrowaniem wyjść funkcji API. Jeśli dla tego pola zostanie określony wskaźnik pusty, przyjmowana jest wartość domyślna struktury MQXEPO.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny, który kwalifikuje kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli kod zakończenia ma wartość MQCC_FAILED, wykonaj następujące czynności:

BŁĄD MQRC_HCONFIG_ERROR

(2280, X'8E8') Podany uchwyt konfiguracji nie jest poprawny. Użyj uchwytu konfiguracji z MQAXP.

MQRC_EXIT_REASON_ERROR

(2377, X' 949 ') Podany powód wywołania funkcji wyjścia jest niepoprawny lub nie jest poprawny dla podanego identyfikatora funkcji wyjścia.

Należy użyć jednego z poprawnych przyczyn wywołania funkcji wyjścia (wartość MQXR_*) lub użyć poprawnego identyfikatora funkcji i kombinacji przyczyny wyjścia. (Patrz [Tabela 205](#) na stronie 1666.)

MQRC_FUNCTION_ERROR

(2281, X'8E9') Podany identyfikator funkcji nie jest poprawny dla przyczyny wyjścia funkcji API. Poniższa tabela zawiera poprawne kombinacje identyfikatorów funkcji i ExitReasons.

Tabela 205. Poprawne kombinacje identyfikatorów funkcji i ExitReasons	
Funkcja	ExitReason
MQXF_INIT MQXF_TERM	MQXR_CONNECTION

Tabela 205. Poprawne kombinacje identyfikatorów funkcji i ExitReasons (kontynuacja)

Funkcja	ExitReason
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CLOSE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_BEGIN MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB Komenda MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_PRZED MQXR_AFTER
MQXF_DATA_CONV_ON_GET	MQXR_PRZED

Problem MQRC_RESOURCE_PROBLEM

(2102, X'836 ') Próba zarejestrowania lub wyrejestrowywania funkcji wyjścia nie powiodła się z powodu problemu z zasobem.

Błąd MQRC_UNEXPECTED_ERROR

(2195, X'893 ') Próba zarejestrowania lub wyrejestrowywania funkcji wyjścia nieoczekiwanie zakończyła się niepowodzeniem.

Błąd MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Niepoprawna nazwa ExitProperties .

BŁĄD MQRC_XEPO_ERROR

(2507, X'09CB') Struktura opcji wyjścia nie jest poprawna.

Wywołanie języka C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Deklaracja dla listy parametrów:

```

MQHCONFIG      Hconfig;          /* Configuration handle */
MQLONG         ExitReason;       /* Exit reason */
MQLONG         Function;        /* Function identifier */
PMQFUNC        EntryPoint;      /* Function entry point */
MQXEPO         ExitOpts;        /* Options that control the action of MQXEP */
MQLONG         CompCode;        /* Completion code */
MQLONG         Reason;          /* Reason code qualifying completion
                                code */

```

Prototyp funkcji C MQXEP

```

void MQXEP (
MQHCONFIG      Hconfig,          /* Configuration handle */
MQLONG         ExitReason,      /* Exit reason */

```

MQLONG	Function,	/* Function identifier */
PMQFUNC	EntryPoint,	/* Function entry point */
PMQXEP0	pExitOpts;	/* Options that control the action of MQXEP */
PMQLONG	pCompCode,	/* Address of completion code */
PMQLONG	pReason);	/* Address of reason code qualifying completion code */

Funkcje wyjścia

Ta sekcja zawiera ogólne informacje pomocne w korzystaniu z wywołań funkcji i opisuje sposób wywoływania poszczególnych funkcji wyjścia.

Te informacje umożliwiają zapoznanie się z ogólnymi regułami dla procedur obsługi wyjścia funkcji API oraz konfigurowanie i czyszczenie środowiska wykonawczego wyjścia.

Ogólne reguły dla procedur wyjścia funkcji API

Podczas wywoływania procedur wyjścia funkcji API mają zastosowanie następujące reguły ogólne:

- We wszystkich przypadkach funkcje wyjścia funkcji API są sterowane przed sprawdzaniem poprawności parametrów wywołania API oraz przed sprawdzeniami zabezpieczeń (w przypadku MQCONN, MQCONNX lub MQOPEN).
- Wartości pól wprowadzonych do procedury zewnętrznej i wyprowadzanych z niej są następujące:
 - W przypadku wejścia do funkcji wyjścia funkcji API *przed* IBM MQ wartość pola może zostać ustawiona przez program użytkowy lub przez poprzednie wywołanie funkcji wyjścia.
 - W przypadku danych wyjściowych z funkcji wyjścia funkcji API *przed* IBM MQ wartość pola może pozostać niezmienną lub ustawić na inną wartość przez funkcję wyjścia.
 - W przypadku wejścia do funkcji wyjścia funkcji API *po* IBM MQ wartość pola może być wartością ustawioną przez menedżer kolejek po przetworzeniu wywołania funkcji API IBM MQ lub może być ustawiona na wartość przez poprzednie wywołanie funkcji wyjścia w łańcuchu funkcji wyjścia.
 - W przypadku wyjścia z funkcji wyjścia wywołania funkcji API *po* IBM MQ wartość pola może pozostać niezmienną lub ustawić na inną wartość przez funkcję wyjścia.
- Funkcje wyjścia muszą komunikować się z menedżerem kolejek przy użyciu pól ExitResponse i ExitResponse2 .
- Pola kodu CompCode i przyczyny informują o powrocie do aplikacji. Funkcje menedżera kolejek i wyjścia mogą ustawiać pola kodu CompCode i Reason code.
- Wywołanie MQXEP zwraca nowe kody przyczyny do funkcji wyjścia, które wywołują MQXEP. Jednak funkcje wyjścia mogą przetłumaczyć te nowe kody przyczyny na wszystkie istniejące kody przyczyn, które mogą być zrozumiane przez istniejące i nowe aplikacje.
- Każdy prototyp funkcji wyjścia ma podobne parametry do funkcji API o dodatkowym poziomie indirection, z wyjątkiem CompCode i Reason.
- Wyjścia interfejsu API mogą wywoływać wywołania MQI (z wyjątkiem wywołania MQDISC), ale te wywołania MQI nie wywołują wyjść funkcji API.

Należy zauważyć, że niezależnie od tego, czy aplikacja znajduje się na serwerze, czy na kliencie, nie można przewidzieć sekwencjonowania wywołań wyjścia funkcji API. Wywołanie BEFORE wyjścia funkcji API może nie być natychmiast śledzone przez wywołanie AFTER .

Po wywołaniu BEFORE może nastąpić kolejne wywołanie BEFORE . Na przykład:

```
PRZED MQCTL
PRZED wywołaniem zwrotnym
PRZED MQPUT
WYWOŁANIE MQPUT
AFTER-wywołanie zwrotne
PO ZMATERIALIZOWANIU MQ
```

lub wersji

PRZED XAOPEN
PRZED MQCONN
PO MQCONN
PO XAOPEN

Na kliencie znajduje się wyjście, które może modyfikować zachowanie wywołania MQCONN lub MQCONN, nazywanych wyjściem PreConnect. Program zewnętrzny PreConnect może zmodyfikować dowolny z parametrów wywołania MQCONN lub MQCONN, w tym nazwę menedżera kolejek. Klient wywołuje to wyjście jako pierwsze, a następnie wywołuje wywołanie MQCONN lub MQCONN. Należy zauważyć, że tylko początkowe wywołanie MQCONN lub MQCONN wywołuje wyjście funkcji API. Kolejne wywołania ponownego połączenia nie mają żadnego efektu.

Środowisko wykonawcze

Ogólnie, wszystkie błędy funkcji wyjścia są przekazywane z powrotem do procedury obsługi wyjścia przy użyciu pól ExitResponse i ExitResponse2 w produkcie MQAXP.

Te błędy z kolei są przekształcane w wartości MQCC_* i MQRC_* i przekazywane z powrotem do aplikacji w polach CompCode i Reason. Jednak wszelkie błędy napotkane w logice procedury obsługi wyjścia są przekazywane z powrotem do aplikacji jako wartości MQCC_* i MQRC_* w polach CompCode i Reason.

Jeśli funkcja MQ_TERM_EXIT zwraca błąd:

- Wywołanie MQDISC zostało już wykonane
- Nie ma innej możliwości napędu *po* funkcji wyjścia MQ_TERM_EXIT (a tym samym wykonania procedury czyszczącej środowiska wykonawczego wyjścia).
- Procedura czyszcząca środowiska wykonawczego wyjścia *nie* została wykonana

Nie można wyładować wyjścia, ponieważ może on nadal być używany. Ponadto inne zarejestrowane wyjścia znajdujące się dalej w łańcuchu wyjścia, dla których *przed* zakończy się pomyślnie, będą kierowane w odwrotnej kolejności.

Konfigurowanie środowiska wykonawczego wyjścia

Podczas przetwarzania jawnego wywołania MQCONN lub MQCONN, logika obsługi wyjścia konfiguruje środowisko wykonawcze wyjścia przed wywołaniem funkcji inicjowania wyjścia (MQ_INIT_EXIT). Konfiguracja środowiska wykonawczego obsługi wyjścia obejmuje ładowanie wyjścia, uzyskiwanie pamięci masowej i inicjowanie struktur parametrów wyjścia. Przydzielany jest również uchwyt konfiguracji wyjścia.

Jeśli w trakcie tej fazy wystąpią błędy, wywołanie MQCONN lub MQCONN kończy się niepowodzeniem z błędem CompCode MQCC_FAILED i jednym z następujących kodów przyczyny:

MQRC_API_EXIT_LOAD_ERROR

Próba załadowania modułu wyjścia funkcji API nie powiodła się.

MQRC_API_EXIT_NOT_FOUND

Nie można znaleźć funkcji wyjścia funkcji API w module wyjścia funkcji API.

MQRC_STORAGE_NOT_AVAILABLE

Próba zainicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API nie powiodła się, ponieważ ilość pamięci masowej była niewystarczająca.

MQRC_API_EXIT_INIT_ERROR

Wystąpił błąd podczas inicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API.

Czyszczenie środowiska wykonawczego wyjścia

Podczas przetwarzania jawnego wywołania MQDISC lub niejawnego żądania odłączenia w wyniku zakończenia aplikacji, logika obsługi wyjścia może wymagać wyczyszczenia środowiska wykonawczego wyjścia po wywołaniu funkcji zakończenia obsługi wyjścia (MQ_TERM_EXIT), jeśli jest ona zarejestrowana.

Czyszczenie środowiska wykonawczego obsługi wyjścia obejmuje zwalnianie pamięci dla struktur parametrów wyjścia, a także usunięcie wszystkich modułów, które zostały wcześniej załadowane do pamięci.

Jeśli w trakcie tej fazy wystąpią błędy, jawne wywołanie MQDISC kończy się niepowodzeniem z błędem CompCode MQCC_FAILED i następującym kodem przyczyny (błędy nie są podświetlone na niejawnych żądaniach rozłączania):

MQRC_API_EXIT_TERM_ERROR

Wystąpił błąd podczas zamykania środowiska wykonawczego dla funkcji wyjścia funkcji API. Wyjście powinno *nie* zwracać żadnego niepowodzenia z tabeli MQDISC przed wywołaniami funkcji wyjścia funkcji API MQ_TERM* lub po niej.

Wyjścia funkcji API dla klientów

Klient korzysta z programu zewnętrznego PreConnect w celu zmodyfikowania zachowania wywołań MQCONN i MQCONNX i nie obsługuje właściwości wyjścia funkcji API.

Wyjście PreConnect

W przypadku klienta wyjście PreConnect może być używane do wyszukiwania definicji kanału z centralnego repozytorium, takiego jak serwer LDAP.

Program obsługi wyjścia PreConnect może także modyfikować dowolny parametr lub wszystkie parametry w wywołaniu MQCONN lub MQCONNX, na przykład nazwę menedżera kolejek.

W przypadku aplikacji klienckich wyjście PreConnect musi zostać wywołane przed wyjściem interfejsu API, ponieważ wyjście MQCONN lub MQCONNX API jest wywoływane tylko wtedy, gdy nazwa menedżera kolejek jest znana i ta nazwa może zostać zmieniona przez program obsługi wyjścia PreConnect .

Należy zauważyć, że tylko początkowe wywołanie MQCONN lub MQCONNX wywołuje wyjście.

Właściwości wyjścia funkcji API

Na serwerze wyjścia funkcji API mogą zarejestrować strukturę MQXEPO w czasie inicjowania. Struktura MQXEPO zawiera pole ExitProperties , które zawiera szczegółowe informacje na temat grupy właściwości, w których jest zainteresowany wyjście. Ma to wpływ na wygenerowanie osobnego uchwytu właściwości komunikatu, które wyjście może manipulować oddzielnie od dowolnego uchwytu właściwości komunikatu aplikacji.

Na kliencie właściwości wyjścia funkcji API nie są obsługiwane. Jeśli podjęta zostanie próba zarejestrowania nazwy grupy właściwości na kliencie, funkcja kończy się niepowodzeniem z kodem przyczyny MQRC_EXIT_PROPS_NOT_SUPPORTED.

Backout-MQ_BACK_EXIT

Funkcja MQ_BACK_EXIT udostępnia funkcję wycofania, która umożliwia wykonywanie *przed* i *po* przetwarzaniu wycofania. Należy użyć identyfikatora funkcji MQXF_BACK z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wycofania wywołania wycofania.

Interfejs do tej funkcji to:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wydź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_BACK_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,          /* Address of connection handle */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

Początek-MQ_BEGIN_EXIT

Funkcja MQ_BEGIN_EXIT udostępnia funkcję wyjścia do wykonania *przed* i *po* przetwarzaniu wywołania MQBEGIN. Użyj identyfikatora funkcji MQXF_BEGIN z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia wywołania MQbegin.

Interfejs do tej funkcji to:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pBeginOptions (PMQBO)-wejście/wyjście

Wskaźnik do rozpoczęcia opcji.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
PMQBO    pBeginOptions;  /* Ptr to begin options */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,          /* Address of connection handle */
PMQBO     ppBeginOptions,  /* Address of ptr to begin options */
PMQLONG   pCompCode,       /* Address of completion code */
PMQLONG   pReason);        /* Address of reason code qualifying completion
                             code */
```

Wywołanie zwrotne-MQ_CALLBACK_EXIT

Funkcja MQ_CALLBACK_EXIT udostępnia funkcję wyjścia w celu wykonania *przed* i *po* przetwarzaniu wywołania zwrotnego. Użyj identyfikatora funkcji MQXF_CALLBACK z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołania funkcji wyjścia wywołania zwrotnego.

Interfejs do tej funkcji to:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBCContext)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia

ExitContext (MQAXC)-wejście/wyjście

Wyjdz ze struktury kontekstu

Hconn (MQHCONN)-wejście/wyjście

Uchwyt połączenia

Opis pMsg

deskryptor komunikatu

pGetMsgOpts

Opcje sterujące działaniem MQGET

pBuffer

Obszar, który ma zawierać dane komunikatu

pMQCBCContext

Dane kontekstowe dla wywołania zwrotnego

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQMD      pMsgDesc;      /* Message descriptor */
PMQGM0     pGetMsgOpts;    /* Options that define the operation of the consumer */
PMQVOID    pBuffer;       /* Area to contain the message data */
PMQCBC     pContext;      /* Context data for the callback */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP      pExitParms;    /* Exit parameter structure */
PMQAXC      pExitContext;  /* Exit context structure */
PMQHCONN    pHconn;       /* Connection handle */
PPMQMD      ppMsgDesc;     /* Message descriptor */
PPMQGM0     ppGetMsgOpts;  /* Options that define the operation of the consumer */
PMQVOID     ppBuffer;      /* Area to contain the message data */
PPMQCBC     ppContext;)    /* Context data for the callback */
```

Użycie notatek

1. Wyjście wywołania zwrotnego jest wywoływane przed wywołaniem konsumenta i po zakończeniu funkcji konsumenta konsumenta. Mimo że struktury MQMD i MQGM0 są przekształcane, zmiana wartości przed wyjściem nie powoduje ponownego napędu pobierania komunikatu z kolejki, ponieważ komunikat został już usunięty z kolejki, która ma zostać dostarczona do funkcji konsumenta.

Zarządzanie funkcjami zwrotnymi-MQ_CB_EXIT

Funkcja MQ_CB_EXIT udostępnia funkcję wyjścia w celu wykonania *przed* i *po* wywołaniu obiektu MQCB. Użyj identyfikatora funkcji MQXF_CB z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia wywołania MQCB.

Interfejs do tej funkcji to:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu

Hconn (MQHCONN)-wejście/wyjście

Uchwyt połączenia

Operacja (MQLONG)-wejście/wyjście

Wartość operacji

pCallbackDesc (PMQCBD)-wejście/wyjście

Deskryptor wywołania zwrotnego

Hobj (MQHOBJ)-wejście/wyjście

Uchwyt obiektu

pMsgDesc (PMQMD)-wejście/wyjście

deskryptor komunikatu

pGetMsgOpts (PMQGMO)-wejście/wyjście

Opcje sterujące działaniem obiektu MQCB

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący CompCode

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
MQLONG     Operation;      /* Operation value. */
MQCBD      pMsgDesc;       /* Callback descriptor. */
MQHOBJ     Hobj;           /* Object handle. */
PMQMD      pMsgDesc;       /* Message descriptor */
PMQGMO     pGetMsgOpts;    /* Options that define the operation of the consumer */
MQLONG     CompCode;       /* Completion code. */
PMQLONG    Reason;         /* Reason code qualifying CompCode. */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_CB_EXIT (
PMQAXP      pExitParms;    /* Exit parameter structure */
PMQAXC      pExitContext;  /* Exit context structure */
PMQHCONN    pHconn;        /* Connection handle */
MQLONG      pOperation;    /* Callback operation */
PMQHOBJ     pHobj;         /* Object handle */
PPMQMD      ppMsgDesc;     /* Message descriptor */
PPMQGMO     ppGetMsgOpts;  /* Options that control the action of MQCB */
PMQLONG     pCompCode;     /* Completion code */
PMQLONG     pReason;       /* Reason code qualifying CompCode */
```

Zamknij-MQ_CLOSE_EXIT

Funkcja MQ_CLOSE_EXIT udostępnia funkcję zamykania wyjścia, która umożliwia wykonanie *przed i po* przetwarzaniu wywołania MQCLOSE. Należy użyć identyfikatora funkcji MQXF_CLOSE z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed i po* wywołania funkcji wyjścia wywołania MQCLOSE.

Interfejs do tej funkcji to:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pHobj (PMQHOBj)-dane wejściowe

Wskaźnik do uchwytu obiektu.

Opcje (MQLONG)-wejście/wyjście

Zamknij opcje.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia ma wartość MQCC_FAILED, funkcja wyjścia może ustawić wartość pola kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
PMQHOBj    pHobj;          /* Ptr to object handle */
MQLONG     Options;        /* Close options */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```

void MQENTRY MQ_CLOSE_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    pHconn,          /* Address of connection handle */
PPMQHOBJ    ppHobj,          /* Address of ptr to object handle */
PMLONG      pOptions,        /* Address of close options */
PMLONG      pCompCode,       /* Address of completion code */
PMLONG      pReason);        /* Address of reason code qualifying
                               completion code */

```

Zatwierdź-MQ_CMIT_EXIT

Funkcja MQ_CMIT_EXIT udostępnia funkcję wyjścia zatwierdzania w celu wykonania *przed* i *po* przetwarzaniu zatwierdzania. Należy użyć identyfikatora funkcji MQXF_CMIT z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* zatwierdzaniu funkcji wyjścia wywołania.

Jeśli operacja zatwierdzania nie powiedzie się, a transakcja zostanie wycofana, wywołanie MQCMIT nie powiedzie się i zostanie wykonane wywołanie MQCC_WARNING i MQRC_BACKED_OUT. Te kody powrotu i przyczyny są przekazywane do dowolnej funkcji wyjścia MQCMIT *po* w celu nadania funkcji wyjścia wskazującym, że jednostka pracy została wycofana.

Interfejs do tej funkcji to:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```

MQAXP      ExitParms;        /* Exit parameter structure */
MQAXC      ExitContext;      /* Exit context structure */
MQHCONN    Hconn;            /* Connection handle */

```



```

MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying completion code */

```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_CMITY_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```

void MQENTRY MQ_CMITY_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,         /* Address of connection handle */
PMQLONG   pCompCode,      /* Address of completion code */
PMQLONG   pReason);       /* Address of reason code qualifying completion
                           code */

```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_GET_EXIT jest używany zarówno dla funkcji wyjścia MQXF_GET, jak i funkcji wyjścia [“MQXF_DATA_CONV_ON_GET”](#) na stronie 1683 .

Dla tych dwóch funkcji wyjścia zdefiniowane są osobne punkty wejścia, tak aby przechwycić *oba* wywołanie MQXEP należy używać dwa razy; dla tego wywołania należy użyć identyfikatora funkcji MQXF_GET.

Ponieważ interfejs MQ_GET_EXIT jest taki sam dla funkcji MQXF_GET i MQXF_DATA_CONV_ON_GET, można użyć jednej funkcji wyjścia dla obu tych funkcji. Pole *Function* w strukturze MQAXP wskazuje, która funkcja obsługi wyjścia została wywołana. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla dwóch obserwacji.

Rozszerzenie połączenia i połączenia-MQ_CONNX_EXIT

Wartość MQ_CONNX_EXIT udostępnia:

- Funkcja wyjścia połączenia w celu wykonania *przed* i *po* przetwarzaniu MQCONN
- Funkcja wyjścia rozszerzenia połączenia służy do wykonywania *przed* i *po* przetwarzaniu MQCONNX

Ten sam interfejs, który został opisany w tym miejscu, jest wywoływany zarówno w przypadku funkcji wyjścia wywołania MQCONN, jak i MQCONNX.

Gdy agent kanału komunikatów (MCA) odpowie na przychodzące połączenie klienta, agent MCA może nawiązać połączenie i wykonać wiele wywołań API IBM MQ , zanim stan klienta będzie w pełni znany. Te wywołania interfejsu API wywołują funkcje wyjścia funkcji API z MQAXC w oparciu o sam program MCA (na przykład w polach UserId i ConnectionName produktu MQAXC).

Gdy agent MCA odpowie na kolejne wywołania interfejsu API klienta przychodzącego, struktura MQAXC jest oparta na kliencie przychodzącym, ustawiając odpowiednio pola UserId i ConnectionName .

Nazwa menedżera kolejek ustawiona przez aplikację w wywołaniu MQCONN lub MQCONNX jest przekazywana do bazowego wywołania połączenia. Każda próba podjęta przez wartość *przed* wartością MQ_CONNX_EXIT w celu zmiany nazwy menedżera kolejek nie ma wpływu.

Użyj identyfikatorów funkcji MQXF_CONN i MQXF_CONNX z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQCONN i MQCONNX.

Wyjście MQ_CONNX_EXIT o nazwie MQXR_BEFORE *nie może* wywoływać żadnych wywołań interfejsu API produktu IBM MQ , ponieważ w tym momencie nie zostało skonfigurowane poprawne środowisko.

Funkcja MQ_CONNX_EXIT nie może wywołać wywołania MQDISC z wywołania wyjścia funkcji API dla połączenia, dla którego jest wywoływana. To ograniczenie ma zastosowanie zarówno do wyjść klienta, jak i do wyjść funkcji API serwera.

Interfejs do MQCONN i MQCONNX jest identyczny:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
&pHconn, &CompCode, &Reason);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

pQMgrNazwa (PMQCHAR)-dane wejściowe

Wskaźnik do nazwy menedżera kolejek dostarczonej w wywołaniu MQCONNX. Wyjście nie może zmienić tej nazwy w wywołaniu MQCONN lub MQCONNX.

pConnectOpts (PMQCNO)-wejście/wyjście

Wskaźnik do opcji, które sterują działaniem wywołania MQCONNX.

Szczegółowe informacje na ten temat zawiera sekcja “MQCNO-opcje połączenia” na stronie 320.

W przypadku funkcji wyjścia MQXF_CONN, pConnectOpts wskazuje na domyślną strukturę opcji łączenia (MQCNO_DEFAULT).

pHconn (PMQHCONN)-dane wejściowe

Wskaźnik do uchwytu połączenia.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
PMQCHAR	pQMgrName;	/* Ptr to Queue manager name */
PMQCNO	pConnectOpts;	/* Ptr to Connection options */
PMQHCONN	pHconn;	/* Ptr to Connection handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
&pHconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```

void MQENTRY MQ_CONNX_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PPMQCHAR    ppQMgrName,      /* Address of ptr to queue manager name */
PPMQCNO     ppConnectOpts,   /* Address of ptr to connection options */
PPMQHCONN   ppHconn,         /* Address of ptr to connection handle */
PMQLONG     pCompCode,       /* Address of completion code */
PMQLONG     pReason);        /* Address of reason code qualifying
                               completion code */

```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_CONNX_EXIT jest używany zarówno dla wywołania MQCONN, jak i wywołania MQCONNX. Jednak dla tych dwóch wywołań zdefiniowane są osobne punkty wejścia. Aby przechwytywać wywołania *both*, wywołanie MQXEP musi być używane co najmniej dwukrotnie-raz z identyfikatorem funkcji MQXF_CONN, a także ponownie z MQXF_CONNX.

Ponieważ interfejs MQ_CONNX_EXIT jest taki sam dla wywołań MQCONN i MQCONNX, dla obu wywołań można użyć pojedynczej funkcji wyjścia; pole *Function* w strukturze MQAXP wskazuje, które wywołanie jest w toku. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla dwóch wywołań.

2. Gdy agent kanału komunikatów (MCA) odpowie na przychodzące połączenie klienta, agent MCA może wywołać pewną liczbę wywołań MQ, zanim stan klienta będzie w pełni znany. Te wywołania programu MQ powodują wywołanie funkcji wyjścia funkcji API z strukturą MQAXC zawierającą dane odnoszące się do agenta MCA, a nie do klienta (na przykład identyfikator użytkownika i nazwa połączenia). Jednak gdy stan klienta jest w pełni znany, kolejne wywołania programu MQ powodują wywołanie funkcji wyjścia funkcji API z odpowiednimi danymi klienta w strukturze MQAXC.

3. Wszystkie funkcje wyjścia MQXR_BEFORE są wywoływane przed przeprowadzaniem sprawdzania poprawności parametrów przez menedżer kolejek. Dlatego parametry mogą być niepoprawne (w tym niepoprawne wskaźniki dla adresów parametrów).

Funkcja MQ_CONNX_EXIT jest wywoływana przed wykonaniem jakichkolwiek sprawdzeń autoryzacji przez menedżer kolejek.

4. Funkcja wyjścia nie może zmieniać nazwy menedżera kolejek określonego w wywołaniu MQCONN lub MQCONNX. Jeśli nazwa jest zmieniana przez funkcję wyjścia, wyniki są niezdefiniowane.
5. Funkcja wyjścia MQXR_BEFORE dla funkcji MQ_CONNX_EXIT nie może wywołać wywołań MQ innych niż MQXEP.

Sterowanie wywołaniem zwrotnym-MQ_CTL_EXIT

Parametr MQ_CTL_EXIT udostępnia funkcję wyjścia żądania subskrypcji, która umożliwia wykonanie *przed* i *po* przetwarzaniu wywołania zwrotnego. Użyj identyfikatora funkcji MQXF_CTL z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania zwrotnego elementu sterującego *przed* i *po*.

Interfejs do tej funkcji to:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

gdzie parametry są następujące:

Hconn (MQHCONN)-wejście/wyjście

Uchwyt połączenia.

Wejście/wyjście operacji (MQLONG)

Operacja jest przetwarzana dla wywołania zwrotnego zdefiniowanego dla podanego uchwytu obiektu

Wejście/wyjście ControlOpts (MQCTLO)

Opcje sterujące działaniem komendy MQCTL

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation being processed */
MQCTLO   ControlOpts;    /* Options that control the action of MQCTL */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_CTL_EXIT (
PMQHCONN pHconn;          /* Address of connection handle */
PMQLONG  pOperation;       /* Address of operation being processed */
PMQCTLO  pControlOpts;     /* Address of options that control the action of MQCTL */
PMQLONG  pCompCode;        /* Address of completion code */
PMQLONG  pReason;          /* Address of reason code qualifying completion code */
```

Rozłącz-MQ_DISC_EXIT

Funkcja MQDISC_EXIT udostępnia funkcję wyjścia odłączania w celu wykonania *przed i po* przetwarzaniu wyjścia MQDISC. Użyj identyfikatora funkcji MQXF_DISC z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed i po* wywołaniu funkcji wyjścia wywołania MQDISC.

Interfejs do tej funkcji to

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
&CompCode, &Reason);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wydź ze struktury kontekstu.

pHconn (PMQHCONN)-dane wejściowe

Wskaźnik do uchwytu połączenia.

W przypadku przed wywołaniem wywołania MQDISC wartość tego pola jest jedną z następujących wartości:

- Uchwyt połączenia zwrócony w wywołaniu MQCONN lub MQCONNX
- Zero, w przypadku środowisk, w których adapter specyficzny dla środowiska nawiąże połączenie z menedżerem kolejek
- Wartość ustawiona przez poprzednie wywołanie funkcji wyjścia

W przypadku wywołania MQDISC wartość tego pola jest równa zeru lub wartości ustawionej przez poprzednie wywołanie funkcji wyjścia.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ukończenie częściowe

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

MQAXP	ExitParms;	/* Exit parameter structure */
MQAXC	ExitContext;	/* Exit context structure */
PMQHCONN	pHconn;	/* Ptr to Connection handle */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,  
              &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_DISC_EXIT (  
  PMQAXP      pExitParms,      /* Address of exit parameter structure */  
  PMQAXC      pExitContext,    /* Address of exit context structure */  
  PPMQHCONN   pHconn,          /* Address of ptr to connection handle */  
  PMQLONG     pCompCode,        /* Address of completion code */  
  PMQLONG     pReason);         /* Address of reason code qualifying  
                                completion code */
```

Pobierz-MQ_GET_EXIT

Funkcja MQ_GET_EXIT udostępnia funkcję pobierania wyjścia, która umożliwia wykonanie *przed* i *po* przetwarzaniu wywołania MQGET.

Istnieją dwa identyfikatory funkcji:

1. Użyj komendy MQXF_GET z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed i po* wywołania funkcji wyjścia wywołania MQGET.
2. Informacje na temat korzystania z identyfikatora funkcji MQXF_DATA_CONV_ON_GET zawiera sekcja [“MQXF_DATA_CONV_ON_GET” na stronie 1683](#).

Interfejs do tej funkcji to:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Hobj (MQHOBJ)-wejście/wyjście

Uchwyt obiektu.

pMsgDesc (PMQMD)-wejście/wyjście

Wskaźnik do deskryptora komunikatu.

pGetMsgOpts (PMQGMO)-wejście/wyjście

Wskaźnik, aby uzyskać opcje komunikatów.

BufferLength (MQLONG)-input/output

Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejście/wyjście

Wskaźnik do buforu komunikatów.

pDataLength (PMQLONG)-input/output

Wskaźnik do pola długości danych.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
```

MQHCONN	Hconn;	/* Connection handle */
MQHOBJ	Hobj;	/* Object handle */
PMQMD	pMsgDesc;	/* Ptr to message descriptor */
PMQPMO	pGetMsgOpts;	/* Ptr to get message options */
MQLONG	BufferLength;	/* Message buffer length */
PMQBYTE	pBuffer;	/* Ptr to message buffer */
PMQLONG	pDataLength;	/* Ptr to data length field */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code */

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_GET_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    pHconn,         /* Address of connection handle */
PMQHOBJ     pHobj,          /* Address of object handle */
PMQMD       ppMsgDesc,       /* Address of ptr to message descriptor */
PPMQGMO     ppGetMsgOpts,    /* Address of ptr to get message options */
PMQLONG     pBufferLength,   /* Address of message buffer length */
PPMQBYTE    ppBuffer,       /* Address of ptr to message buffer */
PPMQLONG    ppDataLength,    /* Address of ptr to data length field */
PMQLONG     pCompCode,       /* Address of completion code */
PMQLONG     pReason);       /* Address of reason code qualifying
                             completion code */
```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_GET_EXIT jest używany zarówno dla funkcji wyjścia MQXF_GET, jak i funkcji wyjścia [“MQXF_DATA_CONV_ON_GET”](#) na stronie 1683 .

Dla tych dwóch funkcji wyjścia zdefiniowane są osobne punkty wejścia, tak aby przechwycić *oba* wywołanie MQXEP należy używać dwa razy; dla tego wywołania należy użyć identyfikatora funkcji MQXF_GET.

Ponieważ interfejs MQ_GET_EXIT jest taki sam dla funkcji MQXF_GET i MQXF_DATA_CONV_ON_GET, można użyć jednej funkcji wyjścia dla obu tych funkcji. Pole *Function* w strukturze MQAXP wskazuje, która funkcja obsługi wyjścia została wywołana. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla dwóch obserwacji.

MQXF_DATA_CONV_ON_GET

Informacje na temat interfejsu można znaleźć w sekcji [MQ_GET_EXIT](#) , a także przykładową deklarację języka C.

Użycie notatek

Jeśli jest zarejestrowany, ten punkt wejścia jest wywoływany, gdy komunikaty docierają do aplikacji, ale przed wystąpieniem konwersji danych. Może to być przydatne, jeśli wyjście interfejsu API wymaga wykonania przetwarzania, takiego jak deszyfrowanie lub dekompresja, zanim komunikat zostanie przekazany do konwersji danych. Wyjście może, jeśli to konieczne, spowodować ominięcie konwersji danych przez zwrócenie komendy MQXCC_SUPPRESS_FUNCTION; więcej informacji na ten temat zawiera sekcja Struktura [MQAXP](#) .

Rejestrowanie dla tego punktu wejścia na kliencie powoduje, że konwersja danych jest wykonywana lokalnie na komputerze klienta. W celu poprawnego działania może być konieczne zainstalowanie programów zewnętrznych konwersji aplikacji na kliencie. Należy pamiętać, że parametr MQXF_DATA_CONV_ON_GET jest również używany do asynchronicznego wykorzystania.

W przypadku korzystania z wywołania `MQ_GET_EXIT` należy użyć komendy `MQXF_DATA_CONV_ON_GET` z przyczyną wyjścia `MQXR_BEFORE` w celu zarejestrowania *przed* funkcją wyjścia konwersji danych `MQGET`.

Nie ma funkcji wyjścia `MQXR_AFTER` dla `MQXF_DATA_CONV_ON_GET`; funkcja wyjścia `MQXR_AFTER` dla `MQXF_GET` udostępnia wymaganą możliwość przetwarzania wyjścia po konwersji danych.

Osobne punkty wejścia są zdefiniowane dla wywołania `MQ_GET_EXIT`, więc aby przechwycić *oba* funkcje wyjścia, należy użyć wywołania `MQXEP` dwa razy. W przypadku tego wywołania należy użyć identyfikatora funkcji `MQXF_DATA_CONV_ON_GET`.

Ponieważ interfejs `MQ_GET_EXIT` jest taki sam dla funkcji `MQXF_GET` i `MQXF_DATA_CONV_ON_GET`, można użyć jednej funkcji wyjścia dla obu tych funkcji. Pole *Function* w strukturze `MQAXP` wskazuje, która funkcja obsługi wyjścia została wywołana. Alternatywnie można użyć wywołania `MQXEP` w celu zarejestrowania różnych funkcji wyjścia dla dwóch obserwacji.

Inicjowanie-MQ_INIT_EXIT

Funkcja `MQ_INIT_EXIT` umożliwia zainicjowanie poziomu połączenia wskazanego przez ustawienie parametru `ExitReason` w produkcie `MQAXP` na wartość `MQXR_CONNECTION`.

Podczas inicjowania należy zwrócić uwagę na następujące informacje:

- Funkcja `MQ_INIT_EXIT` wywołuje komendę `MQXEP` w celu zarejestrowania komend interfejsu API produktu IBM MQ oraz punktów `ENTRY` i `EXIT`, w których jest ona zainteresowana.
- Wyjścia nie muszą przechwytywać wszystkich komend interfejsu API produktu IBM MQ . Funkcje obsługi wyjścia są wywoływane tylko wtedy, gdy zarejestrowano zainteresowanie.
- Pamięć masowa, która ma być używana przez wyjście, może zostać przejęta podczas inicjowania.
- Jeśli wywołanie tej funkcji nie powiedzie się, wywołanie `MQCONN` lub `MQCONNX`, które wywołało tę funkcję, również nie powiedzie się, a parametr `CompCode` i Przyczyna są zależne od wartości pola `ExitResponse` w `MQAXP`.
- Wyjście `MQ_INIT_EXIT` nie może zawierać wywołań funkcji API IBM MQ , ponieważ w tym momencie nie zostało skonfigurowane poprawne środowisko.
- Jeśli operacja `MQ_INIT_EXIT` kończy się niepowodzeniem z błędem `MQXCC_FAILED`, menedżer kolejek zwraca się z wywołania `MQCONN` lub `MQCONNX`, które go wywołało przy użyciu wywołania `MQCC_FAILED` i `MQRC_API_EXIT_ERROR`.
- Jeśli menedżer kolejek napotka błąd podczas inicjowania środowiska wykonawczego funkcji wyjścia funkcji API przed wywołaniem pierwszego elementu `MQ_INIT_EXIT`, menedżer kolejek zwraca się z wywołania `MQCONN` lub `MQCONNX`, które wywołało wartość `MQ_INIT_EXIT` z `MQCC_FAILED` i `MQRC_API_EXIT_INIT_ERROR`.

Interfejs do `MQ_INIT_EXIT` jest następujący:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

CompCode (MQLONG)-wejście/wyjście

Wskaźnik do kodu zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Wskaźnik do kodu przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Kod CompCode i przyczyna zwrócone do aplikacji są zależne od wartości pola ExitResponse w produkcie MQAXP.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_INIT_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQLONG     pCompCode,     /* Address of completion code */
PMQLONG     pReason);      /* Address of reason code qualifying
                             completion code */
```

Użycie notatek

1. Funkcja MQ_INIT_EXIT może wywoływać wywołanie MQXEP w celu zarejestrowania adresów funkcji wyjścia dla określonych wywołań MQ, które mają zostać przechwycone. Nie jest konieczne przechwytywanie wszystkich wywołań produktu MQ lub przechwytywanie wywołań MQXR_BEFORE i MQXR_AFTER. Na przykład pakiet obsługi wyjścia może wybrać przechwytywanie tylko wywołania MQPUT w tabeli MQXR_BEFORE.
2. Pamięć masowa, która ma być używana przez funkcje wyjścia w zestawie wyjścia, może zostać przejęta przez funkcję MQ_INIT_EXIT. Alternatywnie funkcje wyjścia mogą pozyskiwać pamięć masową, gdy są one wywoływane, tak jak i w razie potrzeby. Jednak wszystkie pamięci powinny zostać zwolnione, zanim pakiet obsługi wyjścia zostanie zakończony. Funkcja MQ_TERM_EXIT może zwolnić pamięć masową lub wywołać wcześniej funkcję wyjścia.
3. Jeśli funkcja MQ_INIT_EXIT zwróci wartość MQXCC_FAILED w polu *ExitResponse* komendy MQAXP lub w inny sposób nie powiedzie się, wywołanie MQCONN lub MQCONNx, które spowodowało wywołanie funkcji MQ_INIT_EXIT, również się nie powiedzie, a parametry *CompCode* i *Reason* ustawiają odpowiednie wartości.
4. Funkcja MQ_INIT_EXIT nie może wywołać wywołań MQ innych niż MQXEP.

Zapytaj-MQ_INQ_EXIT

Funkcja MQ_INQ_EXIT udostępnia funkcję uzyskiwania informacji, która umożliwia wykonanie *przed* i *po* przetwarzaniu wywołania MQINQ. Użyj identyfikatora funkcji MQXF_INQ z powodami wyjścia

MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołania funkcji wyjścia wywołania MQINQ.

Interfejs do tej funkcji to:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,  
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,  
             &pCharAttrs, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Hobj (MQHOBJ)-dane wejściowe

Uchwyt obiektu.

SelectorCount (MQLONG)-dane wejściowe

Liczba selektorów

pSelectors (PMQLONG)-wejście/wyjście

Wskaźnik do tablicy wartości selektora.

IntAttrCount (MQLONG)-dane wejściowe

Liczba atrybutów całkowitych.

pIntAttrs (PMQLONG)-wejście/wyjście

Wskaźnik do tablicy wartości atrybutów całkowitych.

CharAttrLength (MQLONG)-input/output

Długość tablicy atrybutów znakowych.

pCharAttrs (PMQCHAR)-wejście/wyjście

Wskaźnik do tablicy atrybutów znaków.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;          /* Exit parameter structure */
```

```

MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQHOBJ   Hobj;           /* Object handle */
MQLONG   SelectorCount; /* Count of selectors */
PMQLONG  pSelectors;     /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;   /* Count of integer attributes */
PMQLONG  pIntAttrs;      /* Ptr to array of integer attributes */
MQLONG   CharAttrLength; /* Length of char attributes array */
PMQCHAR  pCharAttrs;     /* Ptr to character attributes */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */

```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```

MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)

```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```

void MQENTRY MQ_INQ_EXIT (
PMQAXP    pExitParms,    /* Address of exit parameter structure */
PMQAXC    pExitContext,  /* Address of exit context structure */
PMQHCONN  pHconn,        /* Address of connection handle */
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 */
PPMQCHAR  ppCharAttrs,   /* Address of ptr to character attributes array */
PMQLONG   pCompCode,     /* Address of completion code */
PMQLONG   pReason);      /* Address of reason code qualifying completion
                           code */

```

Otwórz-MQ_OPEN_EXIT

Funkcja MQ_OPEN_EXIT udostępnia funkcję otwartej wyjścia, która umożliwia wykonanie *przed* i *po* przetwarzaniu wywołania MQOPEN. Użyj identyfikatora funkcji MQXF_OPEN z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia wywołania MQOPEN.

Interfejs do tej funkcji to

```

MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
             &pHobj, &CompCode, &Reason)

```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wydź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pObjDesc (PMQOD)-input/output

Wskaźnik do deskryptora obiektu.

Opcje (MQLONG)-wejście/wyjście

Opcje otwierania.

pHobj (PMQHOBJS)-dane wejściowe

Wskaźnik do uchwytu obiektu.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ukończenie częściowe

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
PMQOD      pObjDesc;       /* Ptr to object descriptor */
MQLONG     Options;        /* Open options */
PMQHOBJS   pHobj;          /* Ptr to object handle */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
               &pHobj, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_OPEN_EXIT (
PMQAXP      pExitParms,      /* Address of exit parameter structure */
PMQAXC      pExitContext,    /* Address of exit context structure */
PMQHCONN    pHconn,          /* Address of connection handle */
PPMQOD      ppObjDesc,       /* Address of ptr to object descriptor */
PMQLONG     pOptions,        /* Address of open options */
PPMHOBJS    ppHobj,          /* Address of ptr to object handle */
PMQLONG     pCompCode,       /* Address of completion code */
PMQLONG     pReason);        /* Address of reason code qualifying
                               completion code */
```

Put-MQ_PUT_EXIT

Funkcja MQ_PUT_EXIT udostępnia funkcję wyjścia do wykonania *przed* i *po* przetwarzaniu wywołania MQPUT. Należy użyć identyfikatora funkcji MQXF_PUT z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołania funkcji wyjścia wywołania MQPUT.

Interfejs do tej funkcji to:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Hobj (MQHOBJ)-wejście/wyjście

Uchwyt obiektu.

pMsgDesc (PMQMD)-wejść/wyjście

Wskaźnik do deskryptora komunikatu.

pPutMsgOpts (PMQPMO)-wejście/wyjście

Wskaźnik do umieszczenia opcji komunikatu.

BufferLength (MQLONG)-input/output

Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejście/wyjście

Wskaźnik do buforu komunikatów.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
MQHOBJ     Hobj;           /* Object handle */
PMQMD      pMsgDesc;       /* Ptr to message descriptor */
PMQPMO     pPutMsgOpts;    /* Ptr to put message options */
MQLONG     BufferLength;   /* Message buffer length */
PMQBYTE    pBuffer;        /* Ptr to message data */
MQLONG     CompCode;       /* Completion code */
MQLONG     Reason;         /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_PUT_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
```

```

PMQHCONN      pHConn,          /* Address of connection handle */
PMQHOBJS      pHObj,           /* Address of object handle */
PPMQMD        pMsgDesc,        /* Address of ptr to message descriptor */
PPMQPMO       ppPutMsgOpts,     /* Address of ptr to put message options */
PMLONG        pBufferLength,    /* Address of message buffer length */
PMQBYTE       pBuffer,         /* Address of ptr to message buffer */
PMLONG        pCompCode,       /* Address of completion code */
PMLONG        pReason);        /* Address of reason code qualifying
                                completion code */

```

Użycie notatek

- Komunikaty raportu wygenerowane przez menedżer kolejek pomija normalne przetwarzanie wywołań. W związku z tym takie komunikaty nie mogą zostać przechwycone przez funkcję MQ_PUT_EXIT lub funkcji MQPUT1. Jednak komunikaty raportu wygenerowane przez agenta kanału komunikatów są przetwarzane normalnie i dlatego mogą zostać przechwycone przez funkcję MQ_PUT_EXIT lub MQ_PUT1_EXIT. Aby mieć pewność, że przechwytywane są wszystkie komunikaty raportu wygenerowane przez agenta MCA, należy użyć zarówno wartości MQ_PUT_EXIT, jak i MQ_PUT1_EXIT.

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT udostępnia funkcję wyjścia *put one message only* w celu wykonania operacji *before* (przed) i *after* MQPUT1 (po wywołaniu funkcji MQPUT1). Użyj identyfikatora funkcji MQXF_PUT1 z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* MQPUT1 funkcji wyjścia wywołania.

Interfejs do tej funkcji to:

```

MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)

```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwył połączenia.

pObjDesc (PMQOD)-input/output

Wskaźnik do deskryptora obiektu.

pMsgDesc (PMQMD)-wejść/wyjście

Wskaźnik do deskryptora komunikatu.

pPutMsgOpts (PMQPMO)-wejście/wyjście

Wskaźnik do umieszczenia opcji komunikatu.

BufferLength (PMLONG)-input/output

Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejście/wyjście

Wskaźnik do buforu komunikatów.

CompCode (PMLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;         /* Connection handle */
PMQOD      pObjDesc;      /* Ptr to object descriptor */
PMQMD      pMsgDesc;      /* Ptr to message descriptor */
PMQPMO     pPutMsgOpts;    /* Ptr to put message options */
MQLONG     BufferLength;   /* Message buffer length */
PMQBYTE    pBuffer;       /* Ptr to message data */
MQLONG     CompCode;      /* Completion code */
MQLONG     Reason;        /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_PUT1_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN    pHconn,       /* Address of connection handle */
PPMQOD      ppObjDesc,    /* Address of ptr to object descriptor */
PPMQMD      ppMsgDesc,    /* Address of ptr to message descriptor */
PPMQPMO     ppPutMsgOpts,  /* Address of ptr to put message options */
MQLONG      pBufferLength, /* Address of message buffer length */
PPMQBYTE    ppBuffer,     /* Address of ptr to message buffer */
MQLONG      pCompCode,    /* Address of completion code */
MQLONG      pReason);     /* Address of reason code qualifying
                           completion code */
```

Ustaw-MQ_SET_EXIT

Funkcja MQ_SET_EXIT udostępnia funkcję wyjścia zestawu w celu wykonania *przed* i *po* przetwarzaniu wywołania MQSET. Użyj identyfikatora funkcji MQXF_SET z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia wywołania MQSET.

Interfejs do tej funkcji to:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &IntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Hobj (MQHOBJ)-dane wejściowe

Uchwyt obiektu.

SelectorCount (MQLONG)-dane wejściowe

Liczba selektorów

pSelectors (PMQLONG)-wejście/wyjście

Wskaźnik do tablicy wartości selektora.

IntAttrCount (MQLONG)-dane wejściowe

Liczba atrybutów całkowitych.

pIntAttrs (PMQLONG)-wejście/wyjście

Wskaźnik do tablicy wartości atrybutów całkowitych.

CharAttrLength (MQLONG)-input/output

Długość tablicy atrybutów znakowych.

pCharAttrs (PMQCHAR)-wejście/wyjście

Wskaźnik do wartości atrybutu znakowego.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;          /* Exit parameter structure */
MQAXC      ExitContext;        /* Exit context structure */
MQHCONN    Hconn;              /* Connection handle */
MQHOBJ     Hobj;                /* Object handle */
MQLONG     SelectorCount;      /* Count of selectors */
PMQLONG    pSelectors;         /* Ptr to array of attribute selectors */
MQLONG     IntAttrCount;       /* Count of integer attributes */
PMQLONG    pIntAttrs;          /* Ptr to array of integer attributes */
MQLONG     CharAttrLength;     /* Length of char attributes array */
PMQCHAR    pCharAttrs;         /* Ptr to character attributes */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```


Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_SET_EXIT (
PMQAXP    pExitParms,      /* Address of exit parameter structure */
PMQAXC    pExitContext,    /* Address of exit context structure */
PMQHCONN  pHconn,         /* Address of connection handle */
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 */
PPMQCHAR  ppCharAttrs,     /* Address of ptr to character attributes array */
PMQLONG    pCompCode,      /* Address of completion code */
PMQLONG    pReason);       /* Address of reason code qualifying completion
                             code */
```

Status-MQ_STAT_EXIT

Funkcja MQ_STAT_EXIT udostępnia funkcję wyjścia statusu w celu wykonania *przed i po* przetwarzaniu wywołania MQSTAT. Użyj identyfikatora funkcji MQXF_STAT z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed i po* wywołaniu funkcji wyjścia wywołania MQSTAT.

Interfejs do tej funkcji to:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

Typ (MQLONG)-dane wejściowe

Typ informacji o statusie do pobrania.

pStatus (PMQSTS)-dane wyjściowe

Wskaźnik do buforu statusu.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_STAT_EXIT (
PMQAXP    pExitParms,          /* Address of exit parameter structure */
PMQAXC    pExitContext,        /* Address of exit context structure */
PMQHCONN  pHconn,              /* Address of connection handle */
PMQLONG    pType,               /* Address of status type */
PPMQSTS    ppStatus,           /* Address of status buffer */
PMQLONG    pCompCode,          /* Address of completion code */
PMQLONG    pReason);           /* Address of reason code qualifying completion
                                code */
```

Zakończenie-MQ_TERM_EXIT

Funkcja MQ_TERM_EXIT umożliwia zakończenie połączenia na poziomie połączenia, zarejestrowane z identyfikatorem funkcji MQXF_TERM i ExitReason MQXR_CONNECTION. Jeśli jest zarejestrowany, wartość MQ_TERM_EXIT jest wywoływana raz dla każdego żądania rozłączenia.

W ramach zakończenia można zwolnić pamięć masową, która nie jest już wymagana przez wyjście, a także można wykonać dowolną procedurę czyszczenia.

Jeśli operacja MQ_TERM_EXIT zakończy się niepowodzeniem z błędem MQXCC_FAILED, menedżer kolejek zwróci wartość z tabeli MQDISC, która wywołała ten błąd, z wartością MQCC_FAILED i MQRC_API_EXIT_ERROR.

Jeśli menedżer kolejek napotka błąd podczas kończenia środowiska wykonawczego funkcji wyjścia funkcji API po wywołaniu ostatniego obiektu MQ_TERM_EXIT, menedżer kolejek zwraca się z wywołania MQDISC, które wywołało atrybut MQ_TERM_EXIT z MQCC_FAILED i MQRC_API_EXIT_TERM_ERROR.

Interfejs do tej funkcji to:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia ma wartość MQCC_FAILED, funkcja wyjścia może ustawić wartość pola kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Kod CompCode i przyczyna zwrócone do aplikacji są zależne od wartości pola ExitResponse w produkcie MQAXP.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQQLONG    CompCode;       /* Completion code */
MQQLONG    Reason;         /* Reason code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_TERM_EXIT (
    PMQAXP      pExitParms,      /* Address of exit parameter structure */
    PMQAXC      pExitContext,    /* Address of exit context structure */
    PMQQLONG    pCompCode,       /* Address of completion code */
    PMQQLONG    pReason);        /* Address of reason code qualifying
                                completion code */
```

Użycie notatek

1. Funkcja MQ_TERM_EXIT jest opcjonalna. Nie jest konieczne, aby pakiet obsługi wyjścia mógł zarejestrować wyjście z zakończenia, jeśli nie ma możliwości zakończenia przetwarzania.
Jeśli funkcje należące do pakietu wyjścia uzyskują zasoby podczas połączenia, funkcja MQ_TERM_EXIT jest wygodnym punktem, w którym można zwolnić te zasoby, na przykład zwalniając pamięć masową uzyskaną dynamicznie.
2. Jeśli funkcja MQ_TERM_EXIT jest zarejestrowana w momencie wywołania wywołania MQDISC, funkcja wyjścia jest wywoływana po wywołaniu wszystkich funkcji wyjścia MQDISC.
3. Jeśli funkcja MQ_TERM_EXIT zwróci wartość MQXCC_FAILED w polu *ExitResponse* komendy MQAXP lub nie powiedzie się w inny sposób, wywołanie MQDISC, które spowodowało wywołanie metody MQ_TERM_EXIT, również nie powiedzie się, a parametry *CompCode* i *Reason* ustawiają odpowiednie wartości.

Zarejestruj subskrypcję-MQ_SUB_EXIT

Funkcja MQ_SUB_EXIT udostępnia funkcję wyjścia w celu wykonania operacji *przed* i *po* przetwarzaniu ponownej rejestracji subskrypcji. Użyj identyfikatora funkcji MQXF_SUB z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia rejestracji subskrypcji *przed* i *po*.

Interfejs do tej funkcji to:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście/wyjście

Uchwyt połączenia.

pSubDesc-input/output

Tablica selektorów atrybutów.

pHobj -wejście/wyjście

Uchwyt obiektu

pHsub (MQHOBJ) wejścia/wyjścia

Uchwył subskrypcji

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN     Hconn;         /* Connection handle */
PMQSD       pSubDesc;      /* Subscription descriptor */
PMQHOBJS    pHobj;         /* Object Handle */
PMQHOBJS    pHsub;         /* Subscription handle */
MQLONG      CompCode;      /* Completion code */
MQLONG      Reason;        /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
PMQAXP      pExitParms;    /* Exit parameter structure */
PMQAXC      pExitContext;  /* Exit context structure */
PMQHCONN     pHconn;       /* Connection handle */
PPMQSD       ppSubDesc;    /* Subscription descriptor */
PPMHOBJS     ppHobj;       /* Object Handle */
PPMHOBJS     ppHsub;       /* Subscription handle */
PMQLONG      pCompCode;    /* Completion code */
PMQLONG      pReason;      /* Reason code qualifying completion code */
```

Żądanie subskrypcji-MQ_SUBRQ_EXIT

Produkt MQ_SUBRQ_EXIT udostępnia funkcję wyjścia żądania subskrypcji w celu wykonania przetwarzania żądania subskrypcji *przed i po* . Należy użyć identyfikatora funkcji MQXF_SUBRQ z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed i po* funkcjach obsługi wyjścia wywołania żądania subskrypcji.

Interfejs do tej funkcji to:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście/wyjście

Uchwyt połączenia.

pHsub (MQHOBJ) wejścia/wyjścia

Uchwyt subskrypcji

Wejście/wyjście działania (MQLONG)

Działanie

pSubRqOpts (MQSRO), wejście/wyjście

CompCode (MQLONG)-wejście/wyjście

Kod zakończenia, poprawne wartości, dla których są:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie zakończone niepowodzeniem

Przyczyna (MQLONG)-wejście/wyjście

Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kodem zakończenia jest MQCC_OK, jedyną poprawną wartością jest:

MQRC_NONE

(0, x '000') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, to funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN     Hconn;         /* Connection handle */
PMQLONG     pHsub;         /* Subscription handle */
MQLONG      Action;        /* Action */
PMQSRO      pSubRqOpts;    /* Subscription Request Options */
MQLONG      CompCode;      /* Completion code */
MQLONG      Reason;        /* Reason code qualifying completion code */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_SUBRQ_EXIT (
PMQAXP      pExitParms,    /* Address of exit parameter structure */
PMQAXC      pExitContext,  /* Address of exit context structure */
PMQHCONN     pHconn,       /* Address of connection handle */
PPMQHOBJS   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 udostępnia funkcję wyjścia xa_close, która ma zostać wykonana przed i po przetworzeniu xa_close. Należy użyć identyfikatora funkcji MQXF_XACLOSE z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować przed i po funkcji wyjścia wywołania xa_close.

Interfejs do tej funkcji to:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdz ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXa_info (PMQCHAR)-wejście/wyjście

Informacje o menedżerze zasobów specyficzne dla instancji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP      ExitParms;      /* Exit parameter structure */
MQAXC      ExitContext;    /* Exit context structure */
MQHCONN    Hconn;          /* Connection handle */
PMQCHAR    pXa_info;       /* Instance-specific RM info */
MQLONG     Rmid;           /* Resource manager identifier */
MQLONG     Flags;          /* Resource manager options*/
MQLONG     XARetCode;      /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_CLOSE_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN  pHconn,        /* Address of connection handle */
    PPMQCHAR  ppXa_info,     /* Address of instance-specific RM info */
    PMQLONG   pRmid,         /* Address of resource manager identifier */
    PMQLONG   pFlags,        /* Address of resource manager options*/
    PMQLONG   pXARetCode);   /* Address of response from XA call */
```

xa_commit-XA_COMMIT_EXIT

XA_COMMIT_EXIT udostępnia funkcję wyjścia xa_commit do wykonania przed i po przetwarzaniu xa_commit. Użyj identyfikatora funkcji MQXF_XACOMMIT z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować przed i po wywołaniu funkcji wyjścia wywołania xa_commit.

Interfejs do tej funkcji to:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flags (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP    ExitParms;    /* Exit parameter structure */
MQAXC    ExitContext;  /* Exit context structure */
MQHCONN   Hconn;       /* Connection handle */
MQPTR     pXID;        /* Transaction branch ID */
MQLONG    Rmid;        /* Resource manager identifier */
MQLONG    Flags;       /* Resource manager options*/
MQLONG    XARetCode;   /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_COMMIT_EXIT (
    PMQAXP    pExitParms,    /* Address of exit parameter structure */
    PMQAXC    pExitContext,  /* Address of exit context structure */
    PMQHCONN   pHconn,       /* Address of connection handle */
    MQPTR      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

Funkcja XA_COMPLETE_EXIT udostępnia funkcję wyjścia xa_complete do wykonania przed i po przetworzeniu xa_complete. Użyj identyfikatora funkcji MQXF_XACOMPLETE z powodami wyjścia MQXR_BEFORE i MQXR_AFTER w celu zarejestrowania przed i po funkcji wyjścia wywołania funkcji xa_complete.

Interfejs do tej funkcji to:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pHandle (PMQLONG)-wejście/wyjście

Wskaźnik do operacji asynchronicznej.

pRetVal (PMQLONG)-wejście/wyjście

Wartość zwracana operacji asynchronicznej.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
PMQLONG pHandle;      /* Ptr to asynchronous op */
PMQLONG pRetVal;      /* Return value of async op */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetVal, &Rmid, &Flags,
&XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_COMPLETE_EXIT (
    PMQAXP   pExitParms,    /* Address of exit parameter structure */
    PMQAXC   pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn,        /* Address of connection handle */
    PPMQLONG ppHandle,      /* Address of ptr to asynchronous op */
    PPMQLONG ppRetVal,      /* Address of return value of async op */
    PMQLONG  pRmid,         /* Address of resource manager identifier */
    PMQLONG  pFlags,        /* Address of resource manager options*/
    PMQLONG  pXARetCode);   /* Address of response from XA call */
```

xa_end-XA_END_EXIT

Funkcja XA_END_EXIT udostępnia funkcję wyjścia xa_end, która ma zostać wykonana przed i po przetworzeniu xa_end. Użyj identyfikatora funkcji MQXF_XAEND z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania xa_end przed i po zakończeniu operacji.

Interfejs do tej funkcji to:


```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_END_EXIT (
    PMQAXP   pExitParms, /* Address of exit parameter structure */
    PMQAXC   pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn,      /* Address of connection handle */
    PMQPTR   ppXID,       /* Address of transaction branch ID */
    PMQLONG  pRmid,       /* Address of resource manager identifier */
    PMQLONG  pFlags,      /* Address of resource manager options*/
    PMQLONG  pXARetCode); /* Address of response from XA call */
```

xa_forget-XA_FORGET_EXIT

Funkcja XA_FORGET_EXIT udostępnia funkcję wyjścia `xa_forget` do wykonania przed i po przetwarzaniu `xa_forget`. Użyj identyfikatora funkcji `MQXF_XAFORGET` z powodami wyjścia `MQXR_BEFORE` i `MQXR_AFTER` w celu zarejestrowania przed i po funkcji wyjścia wywołania `xa_forget`.

Interfejs do tej funkcji to:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_FORGET_EXIT (
    PMQAXP   pExitParms,    /* Address of exit parameter structure */
    PMQAXC   pExitContext,  /* Address of exit context structure */
    PMQHCONN pHconn,        /* Address of connection handle */
    PMQPTR   ppXID,         /* Address of transaction branch ID */
    PMQLONG  pRmid,         /* Address of resource manager identifier */
    PMQLONG  pFlags,        /* Address of resource manager options*/
    PMQLONG  pXARetCode);   /* Address of response from XA call */
```

xa_open-XA_OPEN_EXIT

XA_OPEN_EXIT udostępnia funkcję wyjścia xa_open, która ma zostać wykonana przed i po przetworzeniu xa_open. Użyj identyfikatora funkcji MQXF_XAOPEN z powodami wyjścia MQXR_BEFORE i MQXR_AFTER w celu zarejestrowania przed i po funkcji wyjścia wywołania xa_open.

Interfejs do tej funkcji to:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXa_info (PMQCHAR)-wejście/wyjście

Informacje o menedżerze zasobów specyficzne dla instancji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
PMQCHAR pXa_info;     /* Instance-specific RM info */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_OPEN_EXIT (
    PMQAXP   pExitParms,    /* Address of exit parameter structure */
    PMQAXC   pExitContext,  /* Address of exit context structure */
    PMQHCONN pHconn,        /* Address of connection handle */
    PPMQCHAR ppXa_info,     /* Address of instance-specific RM info */
    PMQLONG  pRmid,         /* Address of resource manager identifier */
    PMQLONG  pFlags,        /* Address of resource manager options*/
    PMQLONG  pXARetCode);   /* Address of response from XA call */
```

xa_prepare-XA_PREPARE_EXIT

Funkcja XA_PREPARE_EXIT udostępnia funkcję wyjścia xa_prepare, która ma zostać wykonana przed i po przetworzeniu xa_prepare. Użyj identyfikatora funkcji MQXF_XAPREPARE z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować elementy przed i po funkcji wyjścia wywołania xa_prepare.

Interfejs do tej funkcji to:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP ExitParms; /* Exit parameter structure */
MQAXC ExitContext; /* Exit context structure */
MQHCONN Hconn; /* Connection handle */
MQPTR pXID; /* Transaction branch ID */
MQLONG Rmid; /* Resource manager identifier */
MQLONG Flags; /* Resource manager options */
MQLONG XARetCode; /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_PREPARE_EXIT (
    PMQAXP pExitParms, /* Address of exit parameter structure */
    PMQAXC pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn, /* Address of connection handle */
    PMQPTR ppXID, /* Address of transaction branch ID */
    PMQLONG pRmid, /* Address of resource manager identifier */
    PMQLONG pFlags, /* Address of resource manager options */
    PMQLONG pXARetCode); /* Address of response from XA call */
```

xa_recover-XA_RECOVER_EXIT

XA_RECOVER_EXIT udostępnia funkcję wyjścia `xa_recover`, która ma zostać wykonana przed i po przetworzeniu `xa_recover`. Użyj identyfikatora funkcji `MQXF_XARECOVER` z powodami wyjścia `MQXR_BEFORE` i `MQXR_AFTER`, aby zarejestrować przed i po wywołaniu funkcji wyjścia wywołania `xa_recover`.

Interfejs do tej funkcji to:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Count (MQLONG)-wejście/wyjście

Maksymalna liczba identyfikatorów XID w tablicy XID

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP  ExitParms;    /* Exit parameter structure */
MQAXC  ExitContext;  /* Exit context structure */
MQHCONN Hconn;       /* Connection handle */
MQPTR  pXID;         /* Transaction branch ID */
MQLONG Count;        /* Max XIDs in XID array */
MQLONG Rmid;         /* Resource manager identifier */
MQLONG Flags;        /* Resource manager options*/
MQLONG XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_RECOVER_EXIT (
    PMQAXP  pExitParms, /* Address of exit parameter structure */
    PMQAXC  pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn,     /* Address of connection handle */
    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

Funkcja XA_ROLLBACK_EXIT udostępnia funkcję wyjścia xa_rollback, która ma zostać wykonana przed i po przetworzeniu xa_rollback. Użyj identyfikatora funkcji MQXF_XAROLLBACK z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować przed i po wywołaniu funkcji wyjścia wywołania xa_rollback.

Interfejs do tej funkcji to:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
MQPTR   pXID;         /* Transaction branch ID */
MQLONG  Rmid;         /* Resource manager identifier */
MQLONG  Flags;        /* Resource manager options*/
MQLONG  XARetCode;    /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_ROLLBACK_EXIT (
    PMQAXP   pExitParms,    /* Address of exit parameter structure */
    PMQAXC   pExitContext,  /* Address of exit context structure */
    PMQHCONN pHconn,        /* Address of connection handle */
    PMQPTR   ppXID,         /* Address of transaction branch ID */
    PMQLONG  pRmid,         /* Address of resource manager identifier */
    PMQLONG  pFlags,        /* Address of resource manager options*/
    PMQLONG  pXARetCode);   /* Address of response from XA call */
```

xa_start-XA_START_EXIT

Funkcja XA_START_EXIT udostępnia funkcję wyjścia xa_start, która ma zostać wykonana przed i po przetworzeniu xa_start. Użyj identyfikatora funkcji MQXF_XASTART z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować przed i po wywołaniu funkcji wyjścia wywołania xa_start.

Interfejs do tej funkcji to:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwył połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP   ExitParms;    /* Exit parameter structure */
MQAXC   ExitContext;  /* Exit context structure */
MQHCONN Hconn;        /* Connection handle */
```

```

MQPTR   pXID;           /* Transaction branch ID */
MQLONG  Rmid;           /* Resource manager identifier */
MQLONG  Flags;          /* Resource manager options*/
MQLONG  XARetCode;      /* Response from XA call */

```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```

typedef void MQENTRY XA_START_EXIT (
    PMQAXP   pExitParms, /* Address of exit parameter structure */
    PMQAXC   pExitContext, /* Address of exit context structure */
    PMQHCONN pHconn,      /* Address of connection handle */
    MQPTR     ppXID,       /* Address of transaction branch ID */
    MQLONG    pRmid,       /* Address of resource manager identifier */
    MQLONG    pFlags,      /* Address of resource manager options*/
    MQLONG    pXARetCode); /* Address of response from XA call */

```

ax_reg-AX_REG_EXIT

AX_REG_EXIT udostępnia funkcję wyjściową ax_reg do wykonania przed i po przetwarzaniu ax_reg. Użyj identyfikatora funkcji MQXF_AXREG z powodami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcję wyjścia wywołania ax_reg przed i po zakończeniu.

Interfejs do tej funkcji to:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wydź ze struktury kontekstu.

Hconn (MQHCONN)-dane wejściowe

Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście

Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```

MQAXP   ExitParms; /* Exit parameter structure */
MQAXC   ExitContext; /* Exit context structure */
MQPTR    pXID;      /* Transaction branch ID */
MQLONG   Rmid;      /* Resource manager identifier */
MQLONG   Flags;     /* Resource manager options*/
MQLONG   XARetCode; /* Response from XA call */

```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY AX_REG_EXIT (  
    PMQAXP pExitParms, /* Address of exit parameter structure */  
    PMQAXC pExitContext, /* Address of exit context structure */  
    PMQPTR ppXID, /* Address of transaction branch ID */  
    PMQLONG pRmid, /* Address of resource manager identifier */  
    PMQLONG pFlags, /* Address of resource manager options*/  
    PMQLONG pXARetCode); /* Address of response from XA call */
```

ax_unreg-AX_UNREG_EXIT

AX_UNREG_EXIT udostępnia funkcję wyjściową ax_unreg, która ma być wykonana przed i po przetworzeniu ax_unreg. Użyj identyfikatora funkcji MQXF_AXUNREG z powodami wyjścia MQXR_BEFORE i MQXR_AFTER w celu zarejestrowania przed i po funkcjach programu zewnętrznego ax_unreg wywołania wyjścia.

Interfejs do tej funkcji to:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście

Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście

Wydź ze struktury kontekstu.

Rmid (MQLONG)-wejście/wyjście

Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście

Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście

Odpowiedź z wywołania XA.

Wywołanie języka C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP ExitParms; /* Exit parameter structure */  
MQAXC ExitContext; /* Exit context structure */  
MQLONG Rmid; /* Resource manager identifier */  
MQLONG Flags; /* Resource manager options*/  
MQLONG XARetCode; /* Response from XA call */
```

Menedżer kolejek następnie logicznie wywołuje wyjście w następujący sposób:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY AX_UNREG_EXIT (  
    PMQAXP pExitParms, /* Address of exit parameter structure */  
    PMQAXC pExitContext, /* Address of exit context structure */  
    PMQLONG pRmid, /* Address of resource manager identifier */  
    PMQLONG pFlags, /* Address of resource manager options*/  
    PMQLONG pXARetCode); /* Address of response from XA call */
```


Informacje ogólne o wywoływaniu funkcji wyjścia

Ten temat zawiera ogólne wskazówki ułatwiające zaplanowanie wyjść, szczególnie związanych z obsługą błędów i nieoczekiwanych zdarzeń.

Niepowodzenie wyjścia

Jeśli funkcja wyjścia zostanie nieprawidłowo zakończona po destrukcyjnym wyjściu z punktu synchronizacji, wywołanie MQGET, ale przed przekazaniem komunikatu do aplikacji, procedura obsługi wyjścia może wykonać odtwarzanie po awarii i przekazać sterowanie do aplikacji.

W takim przypadku komunikat może zostać utracony. Dzieje się tak, jak to się dzieje, gdy aplikacja nie powiedzie się natychmiast po odebraniu komunikatu z kolejki.

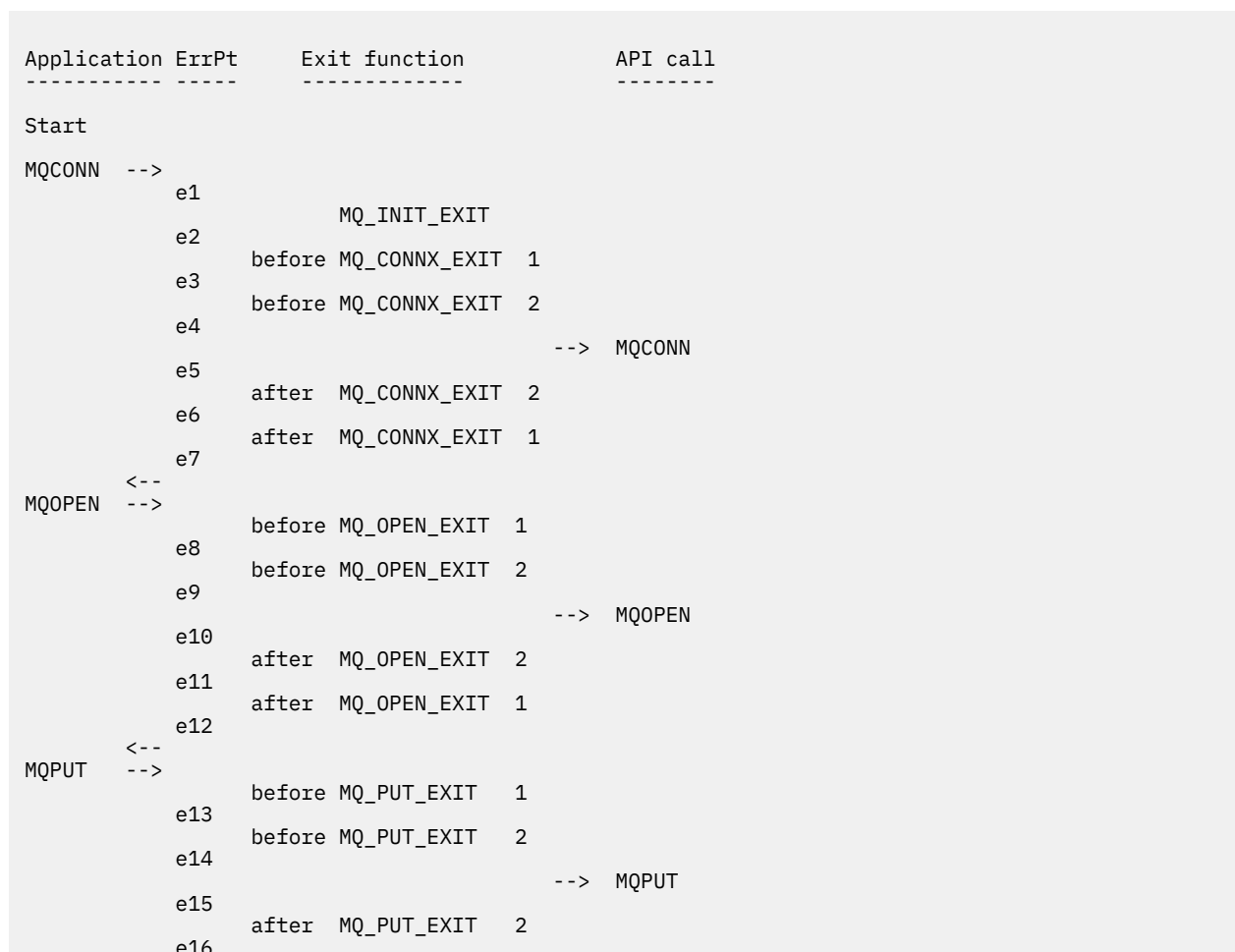
Wywołanie MQGET może zostać zakończone z błędem MQCC_FAILED i MQRC_API_EXIT_ERROR.

Jeśli funkcja obsługi wyjścia wywołania funkcji API *przed* zostanie zakończona nieprawidłowo, procedura obsługi wyjścia może wykonać odtwarzanie po awarii i przekazać sterowanie do aplikacji bez przetwarzania wywołania API. W tym przypadku funkcja wyjścia musi odzyskać wszystkie zasoby, które jest właścicielem.

Jeśli używane programy zewnętrzne są używane, wywołania funkcji API *po* dla dowolnych wyjść wywołania API *przed*, które zostały pomyślnie sterowane, mogą być sterowane samodzielnie. Wywołanie funkcji API może zakończyć się niepowodzeniem z błędem MQCC_FAILED i MQRC_API_EXIT_ERROR.

Przykład obsługi błędów dla funkcji wyjścia

Na poniższym diagramie przedstawiono punkty (e N) w których mogą wystąpić błędy. Jest to tylko przykład, aby pokazać, jak działa działanie wyjść i należy je odczytywać razem z poniższą tabelą. W tym przykładzie dwie funkcje wyjścia są wywoływane zarówno przed, jak i po każdym wywołaniu API, aby pokazać zachowanie za pomocą wyjść tańcuchowych.



```

e17      after MQ_PUT_EXIT  1
MQCLOSE <--
-->
e18      before MQ_CLOSE_EXIT 1
e19      before MQ_CLOSE_EXIT 2
e20
e21      --> MQCLOSE
e22      after MQ_CLOSE_EXIT 2
e23      after MQ_CLOSE_EXIT 1
e24
MQDISC <--
-->
e25      before MQ_DISC_EXIT 1
e26      before MQ_DISC_EXIT 2
e27
e28      --> MQDISC
e29      after MQ_DISC_EXIT 2
e30      after MQ_DISC_EXIT 1
e31
e32
e33
e34
e35
e36
e37
e38
e39
e40
e41
e42
e43
e44
e45
e46
e47
e48
e49
e50
e51
e52
e53
e54
e55
e56
e57
e58
e59
e60
e61
e62
e63
e64
e65
e66
e67
e68
e69
e70
e71
e72
e73
e74
e75
e76
e77
e78
e79
e80
e81
e82
e83
e84
e85
e86
e87
e88
e89
e90
e91
e92
e93
e94
e95
e96
e97
e98
e99
e100
e101
e102
e103
e104
e105
e106
e107
e108
e109
e110
e111
e112
e113
e114
e115
e116
e117
e118
e119
e120
e121
e122
e123
e124
e125
e126
e127
e128
e129
e130
e131
e132
e133
e134
e135
e136
e137
e138
e139
e140
e141
e142
e143
e144
e145
e146
e147
e148
e149
e150
e151
e152
e153
e154
e155
e156
e157
e158
e159
e160
e161
e162
e163
e164
e165
e166
e167
e168
e169
e170
e171
e172
e173
e174
e175
e176
e177
e178
e179
e180
e181
e182
e183
e184
e185
e186
e187
e188
e189
e190
e191
e192
e193
e194
e195
e196
e197
e198
e199
e200
e201
e202
e203
e204
e205
e206
e207
e208
e209
e210
e211
e212
e213
e214
e215
e216
e217
e218
e219
e220
e221
e222
e223
e224
e225
e226
e227
e228
e229
e230
e231
e232
e233
e234
e235
e236
e237
e238
e239
e240
e241
e242
e243
e244
e245
e246
e247
e248
e249
e250
e251
e252
e253
e254
e255
e256
e257
e258
e259
e260
e261
e262
e263
e264
e265
e266
e267
e268
e269
e270
e271
e272
e273
e274
e275
e276
e277
e278
e279
e280
e281
e282
e283
e284
e285
e286
e287
e288
e289
e290
e291
e292
e293
e294
e295
e296
e297
e298
e299
e300
e301
e302
e303
e304
e305
e306
e307
e308
e309
e310
e311
e312
e313
e314
e315
e316
e317
e318
e319
e320
e321
e322
e323
e324
e325
e326
e327
e328
e329
e330
e331
e332
e333
e334
e335
e336
e337
e338
e339
e340
e341
e342
e343
e344
e345
e346
e347
e348
e349
e350
e351
e352
e353
e354
e355
e356
e357
e358
e359
e360
e361
e362
e363
e364
e365
e366
e367
e368
e369
e370
e371
e372
e373
e374
e375
e376
e377
e378
e379
e380
e381
e382
e383
e384
e385
e386
e387
e388
e389
e390
e391
e392
e393
e394
e395
e396
e397
e398
e399
e400
e401
e402
e403
e404
e405
e406
e407
e408
e409
e410
e411
e412
e413
e414
e415
e416
e417
e418
e419
e420
e421
e422
e423
e424
e425
e426
e427
e428
e429
e430
e431
e432
e433
e434
e435
e436
e437
e438
e439
e440
e441
e442
e443
e444
e445
e446
e447
e448
e449
e450
e451
e452
e453
e454
e455
e456
e457
e458
e459
e460
e461
e462
e463
e464
e465
e466
e467
e468
e469
e470
e471
e472
e473
e474
e475
e476
e477
e478
e479
e480
e481
e482
e483
e484
e485
e486
e487
e488
e489
e490
e491
e492
e493
e494
e495
e496
e497
e498
e499
e500
e501
e502
e503
e504
e505
e506
e507
e508
e509
e510
e511
e512
e513
e514
e515
e516
e517
e518
e519
e520
e521
e522
e523
e524
e525
e526
e527
e528
e529
e530
e531
e532
e533
e534
e535
e536
e537
e538
e539
e540
e541
e542
e543
e544
e545
e546
e547
e548
e549
e550
e551
e552
e553
e554
e555
e556
e557
e558
e559
e560
e561
e562
e563
e564
e565
e566
e567
e568
e569
e570
e571
e572
e573
e574
e575
e576
e577
e578
e579
e580
e581
e582
e583
e584
e585
e586
e587
e588
e589
e590
e591
e592
e593
e594
e595
e596
e597
e598
e599
e600
e601
e602
e603
e604
e605
e606
e607
e608
e609
e610
e611
e612
e613
e614
e615
e616
e617
e618
e619
e620
e621
e622
e623
e624
e625
e626
e627
e628
e629
e630
e631
e632
e633
e634
e635
e636
e637
e638
e639
e640
e641
e642
e643
e644
e645
e646
e647
e648
e649
e650
e651
e652
e653
e654
e655
e656
e657
e658
e659
e660
e661
e662
e663
e664
e665
e666
e667
e668
e669
e670
e671
e672
e673
e674
e675
e676
e677
e678
e679
e680
e681
e682
e683
e684
e685
e686
e687
e688
e689
e690
e691
e692
e693
e694
e695
e696
e697
e698
e699
e700
e701
e702
e703
e704
e705
e706
e707
e708
e709
e710
e711
e712
e713
e714
e715
e716
e717
e718
e719
e720
e721
e722
e723
e724
e725
e726
e727
e728
e729
e730
e731
e732
e733
e734
e735
e736
e737
e738
e739
e740
e741
e742
e743
e744
e745
e746
e747
e748
e749
e750
e751
e752
e753
e754
e755
e756
e757
e758
e759
e760
e761
e762
e763
e764
e765
e766
e767
e768
e769
e770
e771
e772
e773
e774
e775
e776
e777
e778
e779
e780
e781
e782
e783
e784
e785
e786
e787
e788
e789
e790
e791
e792
e793
e794
e795
e796
e797
e798
e799
e800
e801
e802
e803
e804
e805
e806
e807
e808
e809
e810
e811
e812
e813
e814
e815
e816
e817
e818
e819
e820
e821
e822
e823
e824
e825
e826
e827
e828
e829
e830
e831
e832
e833
e834
e835
e836
e837
e838
e839
e840
e841
e842
e843
e844
e845
e846
e847
e848
e849
e850
e851
e852
e853
e854
e855
e856
e857
e858
e859
e860
e861
e862
e863
e864
e865
e866
e867
e868
e869
e870
e871
e872
e873
e874
e875
e876
e877
e878
e879
e880
e881
e882
e883
e884
e885
e886
e887
e888
e889
e890
e891
e892
e893
e894
e895
e896
e897
e898
e899
e900
e901
e902
e903
e904
e905
e906
e907
e908
e909
e910
e911
e912
e913
e914
e915
e916
e917
e918
e919
e920
e921
e922
e923
e924
e925
e926
e927
e928
e929
e930
e931
e932
e933
e934
e935
e936
e937
e938
e939
e940
e941
e942
e943
e944
e945
e946
e947
e948
e949
e950
e951
e952
e953
e954
e955
e956
e957
e958
e959
e960
e961
e962
e963
e964
e965
e966
e967
e968
e969
e970
e971
e972
e973
e974
e975
e976
e977
e978
e979
e980
e981
e982
e983
e984
e985
e986
e987
e988
e989
e990
e991
e992
e993
e994
e995
e996
e997
e998
e999
e1000
end

```

Poniższa tabela zawiera listę działań, które mają zostać podjęte w każdym punkcie błędów. Tylko podzbiór punktów błędów został pokryty, ponieważ reguły pokazywane w tym miejscu mogą dotyczyć wszystkich innych. Jest to działania, które określają zamierzone zachowanie w każdym przypadku.

Tabela 206. Błędy wyjścia funkcji API i odpowiednie działania do wykonania		
Err Pt	Opis	Działania
e1	Błąd podczas konfigurowania środowiska.	1. Cofnij konfigurowanie środowiska zgodnie z wymaganiami 2. Napęd bez funkcji wyjścia 3. Niepowodzenie MQCONN z MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	Funkcja MQ_INIT_EXIT kończy się na: • Niepowodzenie MQXCC_FAILED • MQXCC_*	• Dla MQXCC_FAILED: 1. Czyszczenie środowiska 2. Niepowodzenie MQCONN z MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Dla MQXCC_* 1. Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ 2. Czyszczenie środowiska

Tabela 206. Błędy wyjścia funkcji API i odpowiednie działania do wykonania (kontynuacja)		
Err Pt	Opis	Działania
e3	Funkcja <i>Before</i> MQ_CONNX_EXIT 1 kończy się na: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Drive MQ_TERM_EXIT, funkcja - Czyszczenie środowiska - Wywołanie MQCONN nie powiodło się z błędem MQCC_FAILED, MQRC_API_EXIT_ERROR - Dla MQXCC_* - Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ - Napęd MQ_TERM_EXIT napędu, jeśli jest wymagany - Czyszczenie środowiska, jeśli jest to wymagane
e4	Funkcja <i>Before</i> MQ_CONNX_EXIT 2 kończy się na: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 1 - Drive MQ_TERM_EXIT, funkcja - Czyszczenie środowiska - Wywołanie MQCONN nie powiodło się z błędem MQCC_FAILED, MQRC_API_EXIT_ERROR - Dla MQXCC_* - Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 1, jeśli wyjście nie jest pomijalne - Napęd MQ_TERM_EXIT napędu, jeśli jest wymagany - Czyszczenie środowiska, jeśli jest to wymagane
e5	Wywołanie MQCONN nie powiodło się.	- Przekaz wywołania MQCONN CompCode i przyczyny - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 2, jeśli operacja <i>before</i> MQ_CONNX_EXIT 2 zakończyła się pomyślnie, a wyjście nie jest pomijalne. - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 1, jeśli operacja <i>przed</i> MQ_CONNX_EXIT 1 zakończyła się pomyślnie, a wyjście nie jest pomijalne. - Drive MQ_TERM_EXIT, funkcja - Czyszczenie środowiska
e6	Funkcja <i>After</i> MQ_CONNX_EXIT 2 kończy się na: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 1 - Zakończenie wywołania MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR - Dla MQXCC_* - Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ - Napęd <i>po</i> funkcji MQ_CONNX_EXIT 1, jeśli jest to wymagane

Tabela 206. Błędy wyjścia funkcji API i odpowiednie działania do wykonania (kontynuacja)		
Err Pt	Opis	Działania
e7	Funkcja <i>After</i> MQ_CONNX_EXIT 1 kończy się na: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED, pełne wywołanie MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR - W przypadku wartości MQXCC_*, należy użyć wartości parametrów MQXCC_* i MQXR2_*¹.
e8	<i>Przed</i> funkcja MQ_OPEN_EXIT 1 kończy się na: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED, pełne wywołanie MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR - W przypadku wartości MQXCC_*, należy użyć wartości parametrów MQXCC_* i MQXR2_*¹.
e9	<i>Przed</i> funkcją MQ_OPEN_EXIT 2 kończy się: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Napęd <i>po</i> funkcji MQ_OPEN_EXIT 1 - Zakończenie wywołania MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR - W przypadku wartości MQXCC_*, należy użyć wartości parametrów MQXCC_* i MQXR2_*¹.
e10	Wywołanie MQOPEN nie powiodło się	- Przekaz komendy MQOPEN CompCode i przyczyny - Napęd <i>po</i> funkcji MQ_OPEN_EXIT 2, jeśli wyjście nie jest pomijalne - Napęd <i>po</i> funkcji MQ_OPEN_EXIT 1, jeśli wyjście nie jest tłumione i jeśli połączone z nim wyjścia nie są pomijane
e11	<i>Po</i> zakończeniu funkcji MQ_OPEN_EXIT 2 kończy się: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Napęd <i>po</i> funkcji MQ_OPEN_EXIT 1 - Zakończenie wywołania MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR - Dla MQXCC_* - Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ - Napęd <i>po</i> funkcji MQ_OPEN_EXIT 1, jeśli wyjście nie jest pomijalne
e25	<i>Po</i> zakończeniu funkcji MQ_DISC_EXIT 2 kończy się: - Niepowodzenie MQXCC_FAILED - MQXCC_*	- Dla MQXCC_FAILED: - Napęd <i>po</i> funkcji MQ_DISC_EXIT 1 - Drive MQ_TERM_EXIT, funkcja - Czyszczenie środowiska wykonawczego wyjścia - Zakończenie wywołania MQDISC z MQCC_FAILED, MQRC_API_EXIT_ERROR - Dla MQXCC_* - Działanie w odniesieniu do wartości MQXCC_* i MQXR2_*¹ - Drive MQ_TERM_EXIT, funkcja - Czyszczenie środowiska wykonawczego wyjścia

Uwaga:

1. Wartości parametrów MQXCC_* i MQXR2_* i odpowiadające im działania są zdefiniowane w sekcji Jak menedżery kolejek działają funkcje wyjścia.

Pola ExitResponse są ustawione niepoprawnie

Ten temat zawiera informacje o tym, co się stanie, gdy pole ExitResponse jest ustawione na wartość dowolną, ale obsługiwaną wartością.

Jeśli pole ExitResponse jest ustawione na wartość inną niż jedna z obsługiwanych wartości, wówczas zastosowanie mają następujące działania:

- W przypadku funkcji wyjścia funkcji API MQCONN lub MQDISC dla *przed* :
 - Wartość ExitResponse2 jest ignorowana.
 - Nie jest wywoływana żadna dalsza *przed* funkcja wyjścia w łańcuchu wyjścia (jeśli istnieje). Wywołanie funkcji API nie jest wykonywane.
 - Dla wszystkich wyjść *przed* , które zostały pomyślnie wywołane, wyjścia *po* są wywoływane w odwrotnej kolejności.
 - Jeśli rejestracja jest zarejestrowana, funkcje wyjścia z zakończenia dla tych *przed* wywołania MQCONN lub MQDISC w łańcuchu, które zostały pomyślnie wywołane, są kierowane do czyszczenia po tych funkcjach wyjścia.
 - Wywołanie MQCONN lub MQDISC kończy się niepowodzeniem z błędem MQRC_API_EXIT_ERROR.
- W przypadku funkcji wyjścia funkcji API *przed* IBM MQ innych niż MQCONN lub MQDISC:
 - Wartość ExitResponse2 jest ignorowana.
 - W łańcuchu wyjścia (jeśli istnieje) nie są wywoływane żadne dalsze *przed* lub *po* funkcje konwersji danych.
 - Dla wszystkich wyjść *przed* , które zostały pomyślnie wywołane, wyjścia *po* są wywoływane w odwrotnej kolejności.
 - Wywołanie funkcji API IBM MQ nie zostało wydane.
 - Wywołanie funkcji API IBM MQ kończy się niepowodzeniem z błędem MQRC_API_EXIT_ERROR.
- W przypadku funkcji wyjścia funkcji API MQCONN lub MQDISC *po* :
 - Wartość ExitResponse2 jest ignorowana.
 - Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem API, są wywoływane w odwrotnej kolejności.
 - Jeśli rejestracja jest zarejestrowana, funkcje wyjścia zakończenia dla tych *przed* lub *po* funkcji wyjścia MQCONN lub MQDISC w łańcuchu, które zostały pomyślnie wywołane, są kierowane do czyszczenia po wyjściu.
 - Do aplikacji zwracana jest wartość CompCode poważniejszej wartości MQCC_WARNING i CompCode zwróconej przez program obsługi wyjścia.
 - Do aplikacji jest zwracany przyczyna błędu MQRC_API_EXIT_ERROR.
 - Wywołanie funkcji API IBM MQ zostało pomyślnie wydane.
- W przypadku funkcji wyjścia wywołania funkcji API *po* IBM MQ innej niż MQCONN lub MQDISC:
 - Wartość ExitResponse2 jest ignorowana.
 - Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem API, są wywoływane w odwrotnej kolejności.
 - Do aplikacji zwracana jest wartość CompCode poważniejszej wartości MQCC_WARNING i CompCode zwróconej przez program obsługi wyjścia.
 - Do aplikacji jest zwracany przyczyna błędu MQRC_API_EXIT_ERROR.
 - Wywołanie funkcji API IBM MQ zostało pomyślnie wydane.
- W przypadku *przed* konwersji danych w funkcji get exit:
 - Wartość ExitResponse2 jest ignorowana.

- Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem API, są wywoływane w odwrotnej kolejności.
- Komunikat nie zostanie przekształcony, a do aplikacji zostanie zwrócony nieprzekształcony komunikat.
- Do aplikacji zwracana jest wartość CompCode poważniejszej wartości MQCC_WARNING i CompCode zwróconej przez program obsługi wyjścia.
- Do aplikacji jest zwracany przyczyna błędu MQRC_API_EXIT_ERROR.
- Wywołanie funkcji API IBM MQ zostało pomyślnie wydane.

Uwaga: Ponieważ błąd dotyczy wyjścia, lepiej jest zwrócić wartość MQRC_API_EXIT_ERROR, niż zwrócenie wartości MQRC_NOT_CONVERTED.

Jeśli funkcja wyjścia ustawia pole ExitResponse2 na wartość inną niż jedna z obsługiwanych wartości, zamiast niej zostanie przyjęta wartość MQXR2_DEFAULT_CONTINUATION.

Informacje uzupełniające o interfejsie usług instalowalnych

Ta kolekcja tematów zawiera informacje uzupełniające dotyczące instalowalnych usług.

Funkcje i typy danych są wymienione w kolejności alfabetycznej w ramach grupy dla każdego typu usługi.

Sposób wyświetlania funkcji

Sposób dokumentowania funkcji usług instalowalnych.

Dla każdej funkcji znajduje się opis, w tym identyfikator funkcji (dla MQZEP).

Parametry są wyświetlane na liście w kolejności, w jakiej muszą one wystąpić. Wszyscy muszą być obecni.

Po każdej nazwie parametru następuje jego typ danych. Są to elementarne typy danych opisane w [“Elementarne typy danych” na stronie 237](#).

Wywołanie w języku C jest również podane, po opisie parametrów.

MQZ_AUTHENTICATE_USER-Uwierzytelnienie użytkownika

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu uwierzytelnienia użytkownika lub w celu ustawienia pól kontekstu tożsamości. Jest ona wywoływana, gdy kontekst aplikacji użytkownika produktu IBM MQ jest ustanawiany.

Kontekst aplikacji jest ustanawiany podczas wywołań połączenia w punkcie, w którym inicjowany jest kontekst użytkownika aplikacji, oraz w każdym punkcie, w którym zmieniono kontekst użytkownika aplikacji. Za każdym razem, gdy nawiąże się połączenie, informacje o kontekście użytkownika aplikacji są ponownie uzyskiwane w polu *IdentityContext*.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_AUTHENTICATE_USER.

Składnia

MQZ_AUTHENTICATE_USER (*QMgrName* , *SecurityParms* , *ApplicationContext* , *IdentityContext* , *CorrelationPtr* , *ComponentData* , *Kontynuacja* , *CompCode* , *Uzasadnienie*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

SecurityParms

Typ: MQCSP-wejście

Parametry zabezpieczeń. Dane odnoszące się do identyfikatora użytkownika, hasła i typu uwierzytelniania. Jeśli atrybut AuthenticationType struktury MQCSP jest określony jako MQCSP_AUTH_USER_ID_AND_PWD, to zarówno identyfikator użytkownika, jak i hasło są porównywane z równoważnymi polami w parametrze IdentityContext (MQZIC) w celu określenia, czy są one zgodne. Więcej informacji na ten temat zawiera sekcja “MQCSP-parametry zabezpieczeń” na stronie 336.

Podczas wywołania MQCONN MQI ten parametr zawiera wartości NULL lub wartości domyślne.

ApplicationContext

Typ: MQZAC-wejście

Kontekst aplikacji. Dane odnoszące się do aplikacji wywołującej. Szczegółowe informacje na ten temat zawiera sekcja MQZAC-kontekst aplikacji.

Podczas wywoływania wszystkich wywołań MQI MQCONN lub MQCONNX informacje o kontekście użytkownika w strukturze MQZAC są ponownie nabywane.

IdentityContext

Typ: MQZIC-input/output

Kontekst tożsamości. W przypadku danych wejściowych dla funkcji uwierzytelniania użytkownika identyfikuje bieżący kontekst tożsamości. Funkcja uwierzytelniania użytkownika może to zmienić, co oznacza, że menedżer kolejek adoptuje nowy kontekst tożsamości. Więcej informacji na temat struktury MQZIC zawiera sekcja MQZIC-kontekst tożsamości.

CorrelationPtr

Typ: MQPTR-wyjście

Wskaźnik korelacji. Określa adres wszystkich danych korelacji. Wskaźnik ten jest następnie przekazywany do innych wywołań OAM.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Flaga kontynuacji. Możliwe jest określenie następujących wartości:

MQZCI_DEFAULT

Kontynuacja zależna od innych komponentów.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry przekazane do usługi w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCSP     SecurityParms;      /* Security parameters */  
MQZAC     ApplicationContext; /* Application context */  
MQZIC     IdentityContext;    /* Identity context */  
MQPTR     CorrelationPtr;     /* Correlation pointer */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY-sprawdzanie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonywania określonego działania lub działań na określonym obiekcie.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_CHECK_AUTHORITY.

Składnia

MQZ_CHECK_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* ,
ObjectType , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityName

Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa obiektu, którego autoryzacja do obiektu ma zostać sprawdzona. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Nie jest istotne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest ona znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (do której należą wszystkie jednostki). Pusta nazwa jest poprawna i może być używana w ten sposób.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez obiekt EntityName. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie

Typ: MQLONG-wejście

Uprawnienie do sprawdzenia. Jeśli sprawdzana jest jedna autoryzacja, to pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli sprawdzana jest więcej niż jedna autoryzacja, to jest to bitowe LUB odpowiadające im stałe MQZAO_*.

Do korzystania z wywołań MQI stosowane są następujące autoryzacje:

MQZAO_CONNECT

Możliwość korzystania z wywołania MQCONN.

MQZAO_PRZEGLĄDANIE

Możliwość korzystania z wywołania MQGET z opcją przeglądania.

Pozwala to na określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT

Jednostka główna. Możliwość korzystania z wywołania MQGET z opcją wejściową.

Umożliwia to określenie w wywołaniu MQOPEN opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_Q_Q_Q_DEF.

MQZAO_OUTPUT

Możliwość korzystania z wywołania MQPUT.

Pozwala to na określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_ZAPYTANIE_O

Możliwość korzystania z wywołania MQINQ.

Pozwala to na określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET

Możliwość korzystania z wywołania MQSET.

Pozwala to na określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_PASS_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1.

MQZAO_PASS_ALL_CONTEXT

Możliwość przekazania całego kontekstu.

Pozwala to na określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1.

MQZAO_SET_IDENTITY_CONTEXT

Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_SET_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1.

MQZAO_SET_ALL_CONTEXT,

Możliwość ustawienia całego kontekstu.

Pozwala to na określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1.

MQZAO_ALTERNATE_USER_AUTHORITY

Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN oraz opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu komendy MQPUT1.

MQZAO_ALL_MQI

Wszystkie autoryzacje MQI.

Umożliwia to wszystkie autoryzacje.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE

Możliwość tworzenia obiektów o określonym typie.

MQZAO_DELETE

Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY

Możliwość wyświetlania atrybutów określonego obiektu.

ZMIANA MQZAO_CHANGE

Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR

Możliwość usuwania wszystkich komunikatów z określonej kolejki.

MQZAO_AUTORYZACJA

Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL

Możliwość uruchamiania lub zatrzymywania obiektu kanału nasłuchiwanie, usługi lub kanału innego niż klienta oraz możliwości wysyłania pakietów ping do obiektu kanału innego niż klienta.

MQZAO_CONTROL_EXTENDED

Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwej wiadomości na obiekcie kanału innego niż klient.

MQZAO_ALL_ADMIN

Możliwość ustawienia kontekstu tożsamości.

Wszystkie autoryzacje administracyjne, inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do korzystania z interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_ALL

Wszystkie autoryzacje, inne niż MQZAO_CREATE.

MQZAO_NONE

Brak autoryzacji.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

Jeśli wywołanie do komponentu nie powiedzie się (to jest, *CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* to MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek będzie nadal wywoływać inne komponenty, jeśli są jakieś.

Jeśli wywołanie powiedzie się (tj. *CompCode* zwraca wartość MQCC_OK), żadne inne komponenty nie są wywoływane bez względu na to, jakie jest ustawienie *Kontynuacja*.

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie są wywoływane żadne inne komponenty, a błąd jest zwracany do menedżera kolejek. Komponenty nie mają wiedzy o poprzednich wywołaniach, dlatego parametr *Continuation* jest zawsze ustawiony na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY_2 -sprawdzanie uprawnień (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonywania określonego działania lub działań na określonym obiekcie.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_CHECK_AUTHORITY.

Parametr MQZ_CHECK_AUTHORITY_2 jest podobny do komendy MQZ_CHECK_AUTHORITY, ale z parametrem *EntityName* zastąpionym przez parametr *EntityData*.

Składnia

MQZ_CHECK_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do jednostki z autoryzacją do obiektu, który ma zostać sprawdzony. Szczegółowe informacje na ten temat zawiera sekcja “MQZED-deskryptor jednostki” na stronie 1774.

Nie jest istotne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest ona znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (do której należą wszystkie jednostki). Pusta nazwa jest poprawna i może być używana w ten sposób.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nastuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Uprawnienie do sprawdzenia. Jeśli sprawdzana jest jedna autoryzacja, to pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli sprawdzana jest więcej niż jedna autoryzacja, to jest to bitowe LUB odpowiadające im stałe MQZAO_*.

Do korzystania z wywołań MQI stosowane są następujące autoryzacje:

MQZAO_CONNECT

Możliwość korzystania z wywołania MQCONN.

MQZAO_PRZEGLĄDANIE

Możliwość korzystania z wywołania MQGET z opcją przeglądania.

Pozwala to na określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT

Jednostka główna. Możliwość korzystania z wywołania MQGET z opcją wejściową.

Umożliwia to określenie w wywołaniu MQOPEN opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_Q_Q_Q_DEF.

MQZAO_OUTPUT

Możliwość korzystania z wywołania MQPUT.

Pozwala to na określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_ZAPYTANIE_O

Możliwość korzystania z wywołania MQINQ.

Pozwala to na określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET

Możliwość korzystania z wywołania MQSET.

Pozwala to na określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_PASS_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1 .

MQZAO_PASS_ALL_CONTEXT

Możliwość przekazania całego kontekstu.

Pozwala to na określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT

Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_SET_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_ALL_CONTEXT,

Możliwość ustawienia całego kontekstu.

Pozwala to na określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY

Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN oraz opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu komendy MQPUT1 .

MQZAO_ALL_MQI

Wszystkie autoryzacje MQI.

Umożliwia to wszystkie autoryzacje.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE

Możliwość tworzenia obiektów o określonym typie.

MQZAO_DELETE

Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY

Możliwość wyświetlania atrybutów określonego obiektu.

ZMIANA MQZAO_CHANGE

Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR

Możliwość usuwania wszystkich komunikatów z określonej kolejki.

MQZAO_AUTORYZACJA

Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL

Możliwość uruchamiania lub zatrzymywania obiektu kanału nasłuchiwania, usługi lub kanału innego niż klienta oraz możliwości wysyłania pakietów ping do obiektu kanału innego niż klienta.

MQZAO_CONTROL_EXTENDED

Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwej wiadomości na obiekcie kanału innego niż klient.

MQZAO_ALL_ADMIN

Możliwość ustawienia kontekstu tożsamości.

Wszystkie autoryzacje administracyjne, inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do korzystania z interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_ALL

Wszystkie autoryzacje, inne niż MQZAO_CREATE.

MQZAO_NONE

Brak autoryzacji.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptažania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_CHECK_AUTHORITY_2 (QMGrName, &EntityData, EntityType,
```



```
ObjectName, ObjectType, Authority, ComponentData,
&Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED      EntityData;        /* Entity data */
MQLONG     EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG     ObjectType;        /* Object type */
MQLONG     Authority;         /* Authority to be checked */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     Continuation;      /* Continuation indicator set by
                               component */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-sprawdź, czy użytkownik jest uprzywilejowany

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_6 i jest wywoływana przez menedżer kolejek w celu określenia, czy określony użytkownik jest uprzywilejowanym użytkownikiem.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_CHECK_PRIVILEGED.

Składnia

```
MQZ_CHECK_PRIVILEGED( QMgrName , EntityData , EntityType , ComponentData ,
Continuation , CompCode , Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do podmiotu, który ma zostać sprawdzony. Więcej informacji na ten temat zawiera sekcja [“MQZED-deskryptor jednostki”](#) na stronie 1774.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ obiektu określony przez obiekt EntityData. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indyktor kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

Jeśli wywołanie do komponentu nie powiedzie się (to jest, *CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* to MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek będzie nadal wywoływać inne komponenty, jeśli są jakieś.

Jeśli wywołanie powiedzie się (tj. *CompCode* zwraca wartość MQCC_OK), żadne inne komponenty nie są wywoływane bez względu na to, jakie jest ustawienie *Kontynuacja*.

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie są wywoływane żadne inne komponenty, a błąd jest zwracany do menedżera kolejek. Komponenty nie mają wiedzy o poprzednich wywołaniach, dlatego parametr *Continuation* jest zawsze ustawiony na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_PRIVILEGED

(2584, X'A18') Ten użytkownik nie jest uprzywilejowanym identyfikatorem użytkownika.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,  
                      ComponentData, &Continuation,  
                      &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZED     EntityData;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY-kopiowanie wszystkich uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji. Jest on uruchamiany przez menedżer kolejek w celu skopiowania wszystkich autoryzacji, które aktualnie są w stanie, dla obiektu odwołania do innego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_COPY_ALL_AUTHORITY.

Składnia

`MQZ_COPY_ALL_AUTHORITY(QMgrName , RefObjectName , ObjectName , ObjectType ,
ComponentData , Continuation , CompCode , Reason)`

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

NazwaRefObject

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu odniesienia. Nazwa obiektu odniesienia, autoryzacje, dla których mają być skopiowane. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego mają zostać ustawione dostępy. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *RefObjectName* i *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') Nieznany obiekt referencyjny.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,  
                        ComponentData, &Continuation, &CompCode,  
                        &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  RefObjectName;      /* Reference object name */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY-uprawnienie do usuwania

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest uruchamiana przez menedżer kolejek w celu usunięcia wszystkich autoryzacji powiązanych z określonym obiektem.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_AUTHORITY.

Składnia

```
MQZ_DELETE_AUTHORITY( QMgrName , ObjectName , ObjectType , ComponentData ,  
                     Continuation , CompCode , Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego mają zostać usunięte dostępy. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nastuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA-Dane o uprawnieniach Enumerate

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_4 i jest uruchamiana wielokrotnie przez menedżer kolejek w celu pobrania wszystkich danych uprawnień, które są zgodne z kryteriami wyboru określonymi podczas pierwszego wywołania.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_ENUMERATE_AUTHORITY_DATA.

Składnia

```
MQZ_ENUMERATE_AUTHORITY_DATA( QMgrName , StartEnumeration , Filter ,
AuthorityBufferLength , AuthorityBuffer , AuthorityDataLength , ComponentData ,
Continuation , CompCode , Reason )
```

Parametry**QMgrName**

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

StartEnumeration

Typ: MQLONG-wejście

Flaga wskazująca, czy wywołanie może rozpocząć wyliczanie. Wskazuje, czy wywołanie może rozpoczynać wyliczenie danych uprawnień, czy kontynuować wyliczanie danych uprawnień rozpoczętych przez poprzednie wywołanie MQZ_ENUMERATE_AUTHORITY_DATA. Wartość jest jedną z następujących wartości:

MQZSE_START

Początkowe wyliczenie. Wywołanie jest uruchamiane z tą wartością, aby rozpocząć wyliczanie danych uprawnień. Parametr *Filter* określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez te i kolejne wywołania.

MQZSE_CONTINUE

Kontynuuj wyliczanie. Wywołanie jest uruchamiane z tą wartością, aby kontynuować wyliczanie danych uprawnień. Parametr *Filter* jest w tym przypadku ignorowany i może zostać określony jako wskaźnik pusty (kryteria wyboru są określane przez parametr *Filter* określony przez wywołanie, które miało *StartEnumeration* ustawione na wartość MQZSE_START).

Filtr

Typ: MQZAD-wejście

Filtr. Jeśli parametr *StartEnumeration* ma wartość MQZSE_START, *Filter* określa kryteria wyboru, które mają być używane do wybierania danych uprawnień do zwrotu. Jeśli *Filter* jest wskaźnikiem zerowym, nie są używane żadne kryteria wyboru, to znaczy, że zwracane są wszystkie dane uprawnień. Szczegółowe informacje na temat kryteriów wyboru, które można wykorzystać, zawiera sekcja [“MQZAD-dane uprawnień” na stronie 1771](#).

Jeśli parametr *StartEnumeration* ma wartość MQZSE_CONTINUE, *Filter* jest ignorowany i może zostać określony jako wskaźnik pusty.

AuthorityBufferDługość

Typ: MQLONG-wejście

Długość *AuthorityBuffer*. Jest to długość w bajtach parametru *AuthorityBuffer*. Bufor uprawnień musi być wystarczająco duży, aby pomieścić dane, które mają zostać zwrócone.

AuthorityBuffer

Typ: MQZAD-wyjście

Dane uprawnień. Jest to bufor, w którym zwracane są dane uprawnień. Bufor musi być wystarczająco duży, aby pomieścić strukturę MQZAD, strukturę MQZED oraz najdłuższą zdefiniowaną nazwę jednostki i najdłuższą zdefiniowaną nazwę domeny.

Uwaga: Uwaga: Ten parametr jest zdefiniowany jako MQZAD, ponieważ MQZAD zawsze występuje na początku buforu. Jeśli jednak bufor jest zadeklarowany jako zmaterializowana tabela zapytania (MQZAD), bufor będzie zbyt mały-musi być większy niż zmaterializowana tabela zapytania (MQZAD), aby mógł pomieścić nazwy obiektów MQZAD, MQZED oraz jednostki i domeny.

AuthorityDataLength

Typ: MQLONG-wyjście

Długość danych zwracanych w produkcie *AuthorityBuffer*. Jeśli bufor uprawnień jest zbyt mały, parametr *AuthorityDataLength* jest ustawiony na długość wymaganego buforu, a wywołanie zwraca kod zakończenia MQCC_FAILED i kod przyczyny MQRC_BUFFER_LENGTH_ERROR.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_ENUMERATE_AUTHORITY_DATA ma to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_NO_DATA_AVAILABLE

(2379, X'94B') Brak dostępnych danych.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,  
                               &AuthorityBuffer,  
                               &AuthorityDataLength, ComponentData,  
                               &Continuation, &CompCode,  
                               &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQLONG    StartEnumeration;    /* Flag indicating whether call should  
                               start enumeration */
```

MQZAD	Filter;	/* Filter */
MQLONG	AuthorityBufferLength;	/* Length of AuthorityBuffer */
MQZAD	AuthorityBuffer;	/* Authority data */
MQLONG	AuthorityDataLength;	/* Length of data returned in AuthorityBuffer */
MQBYTE	ComponentData[n];	/* Component data */
MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_FREE_USER-użytkownik wolny

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest uruchamiana przez menedżer kolejek w celu zwolnienia powiązanego z nim przydzielonego zasobu.

Jest on uruchamiany, gdy aplikacja zakończyła działanie we wszystkich kontekstach użytkownika, na przykład podczas wywołania MQI MQDISC.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_FREE_USER.

Składnia

MQZ_FREE_USER(QMgrName , FreeParms , ComponentData , Continuation , CompCode , Reason)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

FreeParms

Typ: MQZFP-wejście

Parametry wolne. Struktura zawierająca dane odnoszące się do zasobu, który ma zostać zwolniony. Szczegółowe informacje na ten temat zawiera sekcja [“MQZFP-Wolne parametry”](#) na stronie 1776.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Flaga kontynuacji. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od innych komponentów.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQZFP     FreeParms;          /* Resource to be freed */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY-pobieranie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnień, które jednostka ma do uzyskania dostępu do określonego obiektu, w tym (jeśli jednostka jest jednostką główną) posiadane przez grupy, w których element główny jest elementem. Uprawnienia z profili ogólnych znajdują się w zestawie zwróconych uprawnień.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_AUTHORITY.

Składnia

MQZ_GET_AUTHORITY(QMgrName , EntityName , EntityType , ObjectName ,
ObjectType , Authority , ComponentData , Continuation , CompCode , Reason)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityName

Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa jednostki, której dostęp do obiektu ma zostać pobrany. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego ma zostać pobrany dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Organ jednostki. Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_ *.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptažania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, &Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY_2 -pobranie uprawnień (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnień, które jednostka musi uzyskać w celu uzyskania dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_AUTHORITY.

Parametr MQZ_GET_AUTHORITY_2 jest podobny do wywołania MQZ_GET_AUTHORITY, ale z parametrem *EntityName* zastąpionym przez parametr *EntityData*.

Składnia

```
MQZ_GET_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName ,  
                    ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do obiektu, dla którego ma zostać pobrana autoryzacja do obiektu. Szczegółowe informacje na ten temat zawiera sekcja [“MQZED-deskryptor jednostki” na stronie 1774](#).

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać pobrany organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Organ jednostki. Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_*.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Składnia

MQZ_GET_AUTHORITY_2 (*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Wywołanie C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                    ObjectType, &Authority, ComponentData,
                    &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED      EntityData;        /* Entity data */
MQLONG     EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG     ObjectType;        /* Object type */
MQLONG     Authority;         /* Authority of entity */
MQBYTE     ComponentData[n]; /* Component data */
```


MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_GET_EXPLICIT_AUTHORITY-Uzyskanie jawnego uprawnienia

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnień, które jednostka ma do uzyskania dostępu do określonego obiektu, w tym (jeśli jednostka jest jednostką główną) posiadane przez grupy, w których element główny jest elementem. Uprawnienia z profili ogólnych znajdują się w zestawie zwróconych uprawnień.

W systemie Platformy UNIX dla wbudowanego menedżera uprawnień do obiektów produktu IBM MQ (OAM), zwróconego uprawnienia jest to, że jest ona dostępna tylko przez podstawową grupę podstawową.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_EXPLICIT_AUTHORITY.

Składnia

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityName

Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa obiektu, dla którego ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać pobrany organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Organ jednostki. Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_*.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                             ObjectName, ObjectType, &Authority,
                             ComponentData, &Continuation,
                             &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR12  EntityName;         /* Entity name */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY_2 -pobranie jawnego uprawnienia (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnień, które ma dostęp do określonego obiektu przez nazwaną grupę (ale bez dodatkowego uprawnienia grupy **nobody**), lub uprawnienia, które grupa podstawowa nazwanej nazwy użytkownika ma do dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_EXPLICIT_AUTHORITY.

Parametr MQZ_GET_EXPLICIT_AUTHORITY_2 jest podobny do komendy MQZ_GET_EXPLICIT_AUTHORITY, ale z parametrem *EntityName* zastąpionym parametrem *EntityData*.

Składnia

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do podmiotu, którego uprawnienia do obiektu mają zostać pobrane. Szczegółowe informacje na ten temat zawiera sekcja [“MQZED-deskryptor jednostki” na stronie 1774](#).

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać pobrany organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwanie.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Organ jednostki. Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_ *.

ComponentData

Typ: MQBYTE xComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                               ObjectName, ObjectType, &Authority,
                               ComponentData, &Continuation,
                               &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED     EntityData;         /* Entity data */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY-inicjowanie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest uruchamiana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się, że wywołanie MQZEP będzie możliwe w celu udostępnienia informacji do menedżera kolejek.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_INIT_AUTHORITY.

Składnia

```
MQZ_INIT_AUTHORITY( Hconfig , Options , QMgrName , ComponentDataLength ,
                   ComponentData , Version , CompCode , Reason )
```

Parametry

Konfiguracja Hconfig

Typ: MQHCONFIG-dane wejściowe

Uchwyt konfiguracji. Ten uchwyt reprezentuje określony komponent, który jest inicjowany. Ma ona być używana przez komponent podczas wywoływania menedżera kolejek przy użyciu funkcji MQZEP.

Opcje

Typ: MQLONG-wejście

Opcje inicjowania. Musi to być jedna z następujących wartości:

MQZIO_PRIMARY

Inicjowanie podstawowe.

MQZIO_SECONDARY

Inicjowanie wtórne.

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ComponentDataLength

Typ: MQLONG-wejście

Długość danych komponentu. Długość w bajtach obszaru *ComponentData*. Ta długość jest zdefiniowana w danych konfiguracji komponentu.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Jest on inicjowany dla wszystkich zer przed wywołaniem podstawowej funkcji inicjowania komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i przedstawiane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Wersja

Typ: MQLONG-input/output

Numer wersji. Na wejściu do funkcji inicjowania identyfikuje on najwyższy numer wersji obsługiwany przez menedżera kolejek. Funkcja inicjowania musi to zmienić, jeśli jest to konieczne, do wersji interfejsu, który obsługuje. Jeśli po powrocie menedżer kolejek nie obsługuje wersji zwracanej przez komponent, wywołuje on funkcję MQZ_TERM_AUTHORITY komponentu i nie korzysta z tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1

Wersja 1.

MQZAS_VERSION_2

Wersja 2.

MQZAS_VERSION_3

Wersja 3.

MQZAS_VERSION_4

Wersja 4.

MQZAS_VERSION_5

Wersja 5.

MQZAS_VERSION_6

Wersja 6.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanego powodu.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedociążania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Initialization options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQLONG     ComponentDataLength; /* Length of component data */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     Version;           /* Version number */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_INQUIRE-zapytanie o usługę autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest uruchamiana przez menedżer kolejek w celu wysłania zapytania o obsługiwane funkcje.

W przypadku użycia wielu komponentów usług komponenty usług są wywoływane w kolejności odwrotnej do kolejności, w jakiej zostały zainstalowane.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INQUIRE.

Składnia

```
MQZ_INQUIRE( QMgrName , SelectorCount , Selectors , IntAttrCount , IntAttrs ,
             CharAttrLength , CharAttrs , SelectorReturned , ComponentData , Continuation ,
             CompCode , Reason )
```

Parametry**QMgrName**

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

SelectorCount

Typ: MQLONG-wejście

Liczba selektorów. Liczba selektorów podanych w parametrze *Selectors*.

Wartość musi być z zakresu od 0 do 256.

Selektory

Typ: MQLONGxSelectorCount-input

Tablica selektorów. Każdy selektor identyfikuje wymagany atrybut i musi mieć jedną z następujących wartości:

- MQIACF_INTERFACE_VERSION (liczba całkowita)
- MQIACF_USER_ID_SUPPORT (liczba całkowita)
- MQCACF_SERVICE_COMPONENT (znak)

Selektory mogą być określane w dowolnej kolejności. Liczba selektorów w tablicy jest wskazywana przez parametr *SelectorCount*.

Atrybuty całkowitoliczbowe zidentyfikowane przez selektory są zwracane w parametrze *IntAttrs* w tej samej kolejności, w jakiej są wyświetlane w produkcie *Selectors*.

Atrybuty znakowe zidentyfikowane przez selektory są zwracane w parametrze *CharAttrs* w takiej samej kolejności, w jakiej są wyświetlane *Selectors*.

IntAttrLiczba

Typ: MQLONG-wejście

Liczba atrybutów całkowitoliczbowych podanych w parametrze *IntAttrs*.

Wartość musi być z zakresu od 0 do 256.

IntAttrs

Typ: MQLONG x IntAttrCount-output

Atrybuty całkowite. Tablica atrybutów całkowitoliczbowych. Atrybuty całkowitoliczbowe są zwracane w tej samej kolejności, w jakiej znajdują się odpowiednie selektory całkowite w tablicy *Selectors*.

Licznik znakówCharAttr

Typ: MQLONG-wejście

Długość buforu atrybutów znaków. Długość (w bajtach) parametru *CharAttrs*.

Wartość musi być co najmniej równa sumie długości żądanych atrybutów znakowych. Jeśli atrybuty znaków nie są wymagane, wartość zero jest poprawną wartością.

CharAttrs

Typ: MQLONG x CharAttrCount-output

Bufor atrybutów znaków. Bufor zawierający atrybuty znaków, konkatenowany razem. Atrybuty znakowe są zwracane w takiej samej kolejności, w jakiej znajdują się odpowiednie selektory znaków w tablicy *Selectors*.

Długość buforu jest nadawana przez parametr *CharAttrCount*.

SelectorReturned

Typ: MQLONG x SelectorCount -dane wejściowe

Selektor został zwrócony. Tablica wartości identyfikujących, które atrybuty zostały zwrócone z zestawu żądanych przez selektory w parametrze *Selektory*. Liczba wartości w tej tablicy jest

wskazywana przez parametr *SelectorCount* . Każda wartość w tablicy odnosi się do selektora z odpowiedniej pozycji w tablicy Selektory. Każda wartość jest jedną z następujących wartości:

MQZSL_RETURNED

Atrybut żądany przez odpowiedni selektor w parametrze *Selectors* został zwrócony.

MQZSL_NOT_RETURNED

Atrybut żądany przez odpowiedni selektor w parametrze *Selectors* nie został zwrócony.

Tablica jest inicjowana ze wszystkimi wartościami jako *MQZSL_NOT_RETURNED*. Gdy komponent usługi autoryzacji zwraca atrybut, ustawia odpowiednią wartość w tablicy na wartość *MQZSL_NOT_RETURNED*. Umożliwia to innym komponentom usług autoryzacji, do których jest nawiązywać zapytanie, identyfikowanie atrybutów, które zostały już zwrócone.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

Za mało miejsca dla atrybutów znakowych.

MQRC_INT_COUNT_TOO_SMALL

Zbyt mało miejsca dla atrybutów całkowitych.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR,

Liczba selektorów jest niepoprawna.

MQRC_SELECTOR_ERROR,

Selektor atrybutu jest niepoprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED

Określono zbyt wiele selektorów.

MQRC_INT_ATTR_COUNT_ERROR

Liczba atrybutów całkowitych nie jest poprawna.

MQRC_INT_ATTRS_ARRAY_ERROR

Tablica atrybutów liczb całkowitych nie jest poprawna.

MQRC_CHAR_ATTR_LENGTH_ERROR

Liczba atrybutów znakowych nie jest poprawna.

MQRC_CHAR_ATTRS_ERROR

Łańcuch atrybutów znakowych nie jest poprawny.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
              &IntAttrs, CharAttrLength, &CharAttrs,
              SelectorReturned, ComponentData, &Continuation,
              &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48    QMgrName;           /* Queue manager name */
MQLONG      SelectorCount;      /* Selector count */
MQLONG      Selectors[n];       /* Selectors */
MQLONG      IntAttrCount;       /* IntAttrs count */
MQLONG      IntAttrs[n];        /* Integer attributes */
MQLONG      CharAttrCount;      /* CharAttrs count */
MQLONG      CharAttrs[n];       /* 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-Odśwież wszystkie autoryzacje

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_3 i jest wywoływana przez menedżer kolejek w celu odświeżenia listy autoryzacji przechowywanych wewnętrznie przez komponent.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_REFRESH_CACHE (8L).

Składnia

```
MQZ_REFRESH_CACHE( QMgrName , ComponentData , Continuation , CompCode ,
Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to taki sam efekt jak MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_WARNING:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_REFRESH_CACHE (QMgzName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```

MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n]; /* Component data */
MQLONG    Continuation;     /* Continuation indicator set by
                             component */
MQLONG    CompCode;         /* Completion code */
MQLONG    Reason;           /* Reason code qualifying CompCode */

```

MQZ_SET_AUTHORITY-uprawnienie do ustawiania

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu ustawienia uprawnień, które jednostka musi uzyskać w celu uzyskania dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja przestania wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić przy użyciu tej funkcji.

Składnia

`MQZ_SET_AUTHORITY( QMgrName , EntityName , EntityType , ObjectName , ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )`

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityName

Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa obiektu, dla którego ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nastuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

MQOT_TOPIC

.

Uprawnienie

Typ: MQLONG-wejście

Organ jednostki. Jeśli ustawione jest jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli jest ustawiony więcej niż jeden ośrodek, to pole to jest bitowe OR dla odpowiednich stałych MQZAO_*.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                   ObjectType, Authority, ComponentData,  
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY_2 -uprawnienie do ustawiania (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu ustawienia uprawnień, które jednostka musi uzyskać w celu uzyskania dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja przestania wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić przy użyciu tej funkcji.

MQZ_SET_AUTHORITY_2 jest podobny do MQZ_SET_AUTHORITY, ale z parametrem *EntityName* zastąpionym parametrem *EntityData*.

Składnia

`MQZ_SET_AUTHORITY_2( QMgrName , EntityData , EntityType , ObjectName , ObjectType , Authority , ComponentData , Continuation , CompCode , Reason )`

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do podmiotu, którego uprawnienia do obiektu mają być ustawione. Szczegółowe informacje na ten temat zawiera sekcja [“MQZED-deskryptor jednostki” na stronie 1774](#).

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ jednostki określony przez produkt *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName

Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego ma zostać ustawiony organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami.

Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType

Typ: MQLONG-wejście

Typ obiektu. Typ jednostki określony przez produkt *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwanie.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE**MQOT_TOPIC****Uprawnienie**

Typ: MQLONG-wejście

Organ jednostki. Jeśli ustawione jest jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli jest ustawiony więcej niż jeden ośrodek, to pole to jest bitowe OR dla odpowiednich stałych MQZAO_ *.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                    ObjectType, Authority, ComponentData,
                    &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED     EntityData;         /* Entity data */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority to be checked */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY-kończenie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest uruchamiana przez menedżer kolejek, gdy nie wymaga ona już usług tego komponentu. Funkcja musi wykonać procedurę czyszczącą wymaganą przez komponent.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_TERM_AUTHORITY.

Składnia

```
MQZ_TERM_AUTHORITY( Hconfig , Options , QMgrName , ComponentData , CompCode ,
Reason )
```

Parametry

Konfiguracja Hconfig

Typ: MQHCONFIG-dane wejściowe

Uchwyt konfiguracji. Ten uchwyt reprezentuje określony komponent, który został zakończony. Ma ona być używana przez komponent podczas wywoływania menedżera kolejek przy użyciu funkcji MQZEP.

Opcje

Typ: MQLONG-wejście

Opcje zakończenia. Musi to być jedna z następujących wartości:

MQZTO_PRIMARY

Zakończenie podstawowe.

MQZTO_SECONDARY

Zakończenie wtórne.

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData w wywołaniu MQZ_INIT_AUTHORITY.

Po zakończeniu wywołania MQZ_TERM_AUTHORITY menedżer kolejek odrzuci te dane.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_TERMINATION_FAILED

(2287, X'8FF') Wygaśnienie nie powiodło się z niezdefiniowanego powodu.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;          /* Configuration handle */  
MQLONG     Options;          /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;         /* Completion code */  
MQLONG     Reason;           /* Reason code qualifying CompCode */
```

MQZ_DELETE_NAME-usunięcie nazwy

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek w celu usunięcia pozycji dla podanej kolejki.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_NAME.

Składnia

MQZ_DELETE_NAME(QMgrName , QName , ComponentData , Continuation , CompCode , Reason)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

Nazwa QName

Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać usunięta pozycja. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze długości ComponentData w wywołaniu MQZ_INIT_NAME.

Kontynuacja

Typ: MQLONG-wyjście

Indyktor kontynuacji ustawiony przez komponent. Musi to być jedna z następujących wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

W przypadku komendy **MQZ_DELETE_NAME** menedżer kolejek nie podejmuje próby uruchomienia innego komponentu, bez względu na to, co jest zwracane w parametrze **Continuation**.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_WARNING:

MQRC_UNKNOWN_NAME

(2288, X'8F0') Nie znaleziono nazwy kolejki.

Uwaga: Zwrócenie tego kodu może nie być możliwe, jeśli usługa bazowa odpowiada z powodzeniem dla tej sprawy.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,  
                 &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_NAME-inicjowanie usługi nazw

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się, że wywołanie MQZEP będzie możliwe w celu udostępnienia informacji do menedżera kolejek.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_INIT_NAME.

Składnia

```
MQZ_INIT_NAME( Hconfig , Options , QMgrName , ComponentDataLength ,  
ComponentData , Version , CompCode , Reason )
```

Parametry

Konfiguracja Hconfig

Typ: MQHCONFIG-dane wejściowe

Uchwyt konfiguracji. Ten uchwyt reprezentuje określony komponent, który jest inicjowany. Ma ona być używana przez komponent podczas wywoływania menedżera kolejek przy użyciu funkcji MQZEP.

Opcje

Typ: MQLONG-wejście

Opcje inicjowania. Musi to być jedna z następujących wartości:

MQZIO_PRIMARY

Inicjowanie podstawowe.

MQZIO_SECONDARY

Inicjowanie wtórne.

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ComponentDataLength

Typ: MQLONG-wejście

Długość danych komponentu. Długość w bajtach obszaru *ComponentData*. Ta długość jest zdefiniowana w danych konfiguracji komponentu.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Jest on inicjowany dla wszystkich zer przed wywołaniem podstawowej funkcji inicjowania komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i przedstawiane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Wersja

Typ: MQLONG-input/output

Numer wersji. Na wejściu do funkcji inicjowania identyfikuje on najwyższy numer wersji obsługiwany przez menedżera kolejek. Funkcja inicjowania musi to zmienić, jeśli jest to konieczne, do wersji interfejsu, który obsługuje. Jeśli po powrocie menedżer kolejek nie obsługuje wersji zwracanej przez komponent, wywoła ona funkcję MQZ_TERM_NAME komponentu i nie korzysta z tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1

Wersja 1.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanego powodu.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,  
               ComponentData, &Version, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG    Hconfig;           /* Configuration handle */  
MQLONG       Options;           /* Initialization options */  
MQCHAR48     QMgrName;         /* Queue manager name */  
MQLONG       ComponentDataLength; /* Length of component data */  
MQBYTE       ComponentData[n]; /* Component data */  
MQLONG       Version;          /* Version number */  
MQLONG       CompCode;         /* Completion code */  
MQLONG       Reason;           /* Reason code qualifying CompCode */
```

MQZ_INSERT_NAME-wstaw nazwę

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek w celu wstawienia pozycji dla określonej kolejki, zawierającej nazwę menedżera kolejek, do którego należy kolejka. Jeśli kolejka jest już zdefiniowana w usłudze, wywołanie nie powiedzie się.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INSERT_NAME.

Składnia

```
MQZ_INSERT_NAME( QMgrName , QName , ResolvedQMgrName , ComponentData ,  
                Continuation , CompCode , Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

Nazwa QName

Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać wstawiony wpis. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

ResolvedQMgrNazwa

Typ: MQCHAR48 -dane wejściowe

Rozstrzygnięta nazwa menedżera kolejek. Nazwa menedżera kolejek, do którego jest rozstrzygana kolejka. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i przedstawiane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_NAME.

Kontynuacja

Typ: MQLONG-input/output

Indykator kontynuacji ustawiony przez komponent. W przypadku nazwy MQZ_INSERT_NAME menedżer kolejek nie podejmuje próby uruchomienia innego komponentu, niezależnie od tego, czy jest zwracany w parametrze *Continuation*.

Obsługiwane są następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS

(2290, X'8F2') Obiekt kolejki już istnieje.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,  
                 &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  QName;              /* Queue name */  
MQCHAR48  ResolvedQMgrName;   /* Resolved queue manager name */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_LOOKUP_NAME-nazwa wyszukiwania

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek w celu pobrania nazwy menedżera kolejek, do którego należy dany menedżer kolejek, dla określonej kolejki.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_LOOKUP_NAME.

Składnia

```
MQZ_LOOKUP_NAME( QMgrName , QName , ResolvedQMgrName , ComponentData ,  
Continuation , CompCode , Reason )
```

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

Nazwa QName

Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać rozstrzygnięty wpis. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

ResolvedQMgrNazwa

Typ: MQCHAR48 -dane wyjściowe

Rozstrzygnięta nazwa menedżera kolejek. Jeśli działanie funkcji zakończy się pomyślnie, jest to nazwa menedżera kolejek, który jest właścicielem kolejki.

Nazwa zwracana przez komponent usługi musi być dopełniona z prawej strony znakami odstępu do pełnej długości parametru; nazwa nie może być zakończona znakiem o kodzie zero lub zawierać początkowe lub osadzone odstępy.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i przedstawiane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_NAME.

Kontynuacja

Typ: MQLONG-wyjście

Indyktor kontynuacji ustawiony przez komponent. W przypadku tabeli MQZ_LOOKUP_NAME menedżer kolejek określa, czy ma być uruchamiany inny komponent usługi nazw w następujący sposób:

- Jeśli parametr *CompCode* ma wartość MQCC_OK, nie są uruchamiane żadne dalsze komponenty, bez względu na wartość zwracaną w sekcji *Kontynuacja*.
- Jeśli *CompCode* nie jest MQCC_OK, uruchamiany jest kolejny komponent, o ile *Continuation* nie jest typu MQZCI_STOP.

Obsługiwane są następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_Q_NAME

(2288, X'8F0') Nie znaleziono nazwy kolejki.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_LOOKUP_NAME (QMGrName, QName, ResolvedQMGrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```

MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQCHAR48  ResolvedQMGrName;  /* Resolved queue manager name */
MQBYTE    ComponentData[n]; /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                             component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;           /* Reason code qualifying CompCode */

```

MQZ_TERM_NAME-Zakończenie usługi nazw

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek, gdy nie wymaga ona już usług tego komponentu. Funkcja musi wykonać procedurę czyszczącą wymaganą przez komponent.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_TERM_NAME.

Składnia

MQZ_TERM_NAME(*Hconfig* , *Options* , *QMGrName* , *ComponentData* , *CompCode* , *Reason*)

Parametry

Konfiguracja *Hconfig*

Typ: MQHCONFIG-dane wejściowe

Uchwyt konfiguracji. Ten uchwyt reprezentuje określony komponent, który został zakończony. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje

Typ: MQLONG-wejście

Opcje zakończenia. Musi to być jedna z następujących wartości:

MQZTO_PRIMARY

Zakończenie podstawowe.

MQZTO_SECONDARY

Zakończenie wtórne.

QMGrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ComponentData

Typ: MQBYTE x ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i przedstawiane przy następnym wywołaniu jednej z tych funkcji komponentu.

Dane komponentu znajdują się w pamięci współużytkowanej dostępnej dla wszystkich procesów.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_NAME.

Gdy wywołanie MQZ_TERM_NAME zostało zakończone, menedżer kolejek odrzuci te dane.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_TERMINATION_FAILED

(2287, X'8FF') Wygaśnienie nie powiodło się z niezdefiniowanego powodu.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,  
                &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC-kontekst aplikacji

Struktura MQZAC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru *ApplicationContext*. Ten parametr określa dane związane z aplikacją wywołującą.

[Tabela 1](#) podsumowuje pola w strukturze.

Tabela 207. Pola w MQZAC	
Pole	Opis
StrucId	Identyfikator struktury
Wersja	Numer wersji struktury
ProcessId	Identyfikator procesu
ThreadId	Identyfikator wątku
ApplName	Nazwa aplikacji
UserID	Identyfikator użytkownika

Tabela 207. Pola w MQZAC (kontynuacja)

Pole	Opis
<u>Identyfikator użytkownikaEffectiveUser</u>	Efektywny identyfikator użytkownika
<u>Środowisko</u>	Środowisko
<u>CallerType</u>	Typ programu wywołującego
<u>AuthenticationType</u>	Typ uwierzytelniania
<u>BindType</u>	Typ powiązania

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZAC_STRUC_ID

Identyfikator struktury kontekstu aplikacji.

Dla języka programowania C jest również zdefiniowana stała zmienna MQZAC_STRUC_ID_ARRAY; ma taką samą wartość jak MQZAC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQZAC_VERSION_1

Struktura kontekstu aplikacji Version-1 . Stała MQZAC_CURRENT_VERSION określa numer wersji bieżącej wersji.

ProcessId

Typ: MQPID-wejście

Identyfikator procesu aplikacji.

ThreadId

Typ: MQTID-wejście

Identyfikator wątku aplikacji.

ApplName

Typ: MQCHAR28 -dane wejściowe

Nazwa aplikacji.

UserID

Typ: MQCHAR12 -dane wejściowe

Identyfikator użytkownika. W polu Systemy UNIX to pole określa rzeczywisty identyfikator użytkownika aplikacji. W polu Windows to pole określa identyfikator użytkownika aplikacji.

Identyfikator użytkownikaEffectiveUser

Typ: MQCHAR12 -dane wejściowe

Efektywny identyfikator użytkownika. W polu Systemy UNIX to pole określa efektywny identyfikator użytkownika aplikacji. W Windows to pole jest puste.

Środowisko

Typ: MQLONG-wejście

Środowisko. To pole określa środowisko, z którego połączenie zostało wykonane. Pole jest jedną z następujących wartości:

MQXE_COMMAND_SERVER

Serwer komend

MQXE_MQSCInterpreter komendy **runmqsc****MQXE_MCA**

Agent kanału komunikatów MQXE_OTHER

MQXE_INNY

Niezdefiniowane środowisko

CallerType

Typ: MQLONG-wejście

Typ programu wywołującego. To pole określa typ programu, który wywołał wywołanie. Pole jest jedną z następujących wartości:

MQXACT_EXTERNAL

Wywołanie jest zewnętrzne w stosunku do menedżera kolejek.

MQXACT_INTERNAL

Wywołanie jest wewnętrzne dla menedżera kolejek.

AuthenticationType

Typ: MQLONG-wejście

Typ uwierzytelniania. To pole określa typ wykonywanego uwierzytelniania. Pole jest jedną z następujących wartości:

MQZAT_INITIAL_CONTEXT

Wywołanie uwierzytelniania jest spowodowane zainicjowaniem kontekstu użytkownika. Ta wartość jest używana podczas wywołania MQCONN lub MQCONNX.

MQZAT_CHANGE_CONTEXT,

Wywołanie uwierzytelniania jest spowodowane zmianą kontekstu użytkownika. Ta wartość jest używana, gdy agent MCA zmienia kontekst użytkownika. Temat nadrzędny: MQZAC-

BindType

Typ: MQLONG-wejście

Typ powiązania. To pole określa typ powiązania, który ma być używany. Pole jest jedną z następujących wartości:

MQCNO_FASTPATH_BINDING

Powiązanie krótkiej ścieżki.

MQCNO_SHARED_BINDING

Powiązanie współużytkowane.

MQCNO_ISOLATED_BINDING

Powiązanie izolowane.

Deklaracja C

Zadeklaruj pola struktury w następujący sposób:

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */
    MQCHAR28   ApplName;         /* Application name */
    MQCHAR12   UserID;           /* User identifier */
    MQCHAR12   EffectiveUserID;   /* Effective user identifier */
    MQLONG     Environment;      /* Environment */
    MQLONG     CallerType;       /* Caller type */
    MQLONG     AuthenticationType; /* Authentication type */
}
```

```

    MQLONG    BindType;          /* Bind type */
};

```

MQZAD-dane uprawnień

Struktura MQZAD jest używana w wywołaniu MQZ_ENUMERATE_AUTHORITY_DATA dla dwóch parametrów, jednego wejścia i jednego wyjścia.

- Parametr MQZAD jest używany dla parametru **Filter**, który jest wprowadzany do wywołania. Ten parametr określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez wywołanie.
- Produkt MQZAD jest również używany w przypadku parametru **AuthorityBuffer**, który jest wyjściem wywołania. Ten parametr określa autoryzacje dla jednej kombinacji nazwy profilu, typu obiektu i obiektu.

Tabela 1. podsumowuje pola w strukturze.

Tabela 208. Pola w tabeli MQZAD	
Pole	Opis
StrucId	Identyfikator struktury
Wersja	Numer wersji struktury
ProfileName	Nazwa profilu
ObjectType	Typ obiektu
Uprawnienie	Uprawnienie
EntityDataPtr	Wskaźnik do danych jednostki
EntityType	Typ jednostki
Opcje	Opcje

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZAD_STRUC_ID

Identyfikator struktury danych uprawnień.

Dla języka programowania C zdefiniowana jest również stała MQZAD_STRUC_ID_ARRAY; ma taką samą wartość jak MQZAD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQZAD_VERSION_1

Struktura kontekstu aplikacji Version-1. Stała MQZAD_CURRENT_VERSION określa numer wersji bieżącej wersji.

Następująca stała określa numer wersji bieżącej wersji:

MQZAD_CURRENT_VERSION

Bieżąca wersja struktury danych uprawnień.

ProfileName

Typ: MQCHAR48 -dane wejściowe

Nazwa profilu.

W przypadku parametru **Filter** to pole jest nazwą profilu, dla którego wymagane są dane uprawnień. Jeśli nazwa jest całkowicie pusta aż do końca pola lub pierwszego znaku o kodzie zero, zwracane są dane o uprawnieniach dla wszystkich nazw profili.

W przypadku parametru **AuthorityBuffer** to pole jest nazwą profilu, który jest zgodny z podanymi kryteriami wyboru.

ObjectType

Typ: MQLONG-wejście

Typ obiektu.

W przypadku parametru **Filter** to pole jest typem obiektu, dla którego wymagane są dane uprawnień. Jeśli wartością jest MQOT_ALL, zwracane są dane o uprawnieniach dla wszystkich typów obiektów.

W przypadku parametru **AuthorityBuffer** to pole jest typem obiektu, do którego ma zastosowanie profil identyfikowany przez parametr **ProfileName**.

Wartość ta jest jedną z następujących wartości: dla parametru **Filter** wartość MQOT_ALL jest również poprawna:

MQOT_AUTH_INFO

Informacje uwierzytelniające

MQOT_CHANNEL

Kanał

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta

MQOT_LISTENER

Program nasłuchujący

MQOT_NAMELIST,

Lista nazw

MQOT_PROCESS

Definicja procesu

Kolejka MQOT_Q

Kolejka

MQOT_Q_MGR

Menedżer kolejek

Usługa MQOT_SERVICE

Usługa

Uprawnienie

Typ: MQLONG-wejście

Uprawnienie.

W przypadku parametru **Filter** to pole jest ignorowane.

W przypadku parametru **AuthorityBuffer** to pole reprezentuje autoryzacje, które jednostka ma do obiektów zidentyfikowanych przez produkty **ProfileName** i **ObjectType**. Jeśli jednostka ma tylko jedno uprawnienie, to pole jest równe odpowiedniej wartości autoryzacji (stała MQZAO_*). Jeśli jednostka ma więcej niż jeden organ, to pole jest bitowe OR odpowiednich stałych MQZAO_*.

EntityDataPtr

Typ: PMQZED-wejście

Adres struktury MQZED identyfikujący obiekt.

W przypadku parametru **Filter** to wskazuje na strukturę MQZED, która identyfikuje jednostkę, dla której wymagane są dane uprawnień. Jeśli parametr **EntityDataPtr** jest wskaźnikiem wartości NULL, zwracane są dane uprawnień dla wszystkich obiektów.

W przypadku parametru **AuthorityBuffer** pole to wskazuje na strukturę MQZED, która identyfikuje jednostkę, dla której zwracane są dane uprawnień.

EntityType

Typ: MQLONG-wejście

Typ jednostki.

W przypadku parametru **Filter** to pole określa typ jednostki, dla której wymagane są dane uprawnień. Jeśli wartością jest MQZAET_NONE, zwracane są dane o uprawnieniach dla wszystkich typów jednostek.

W przypadku parametru **AuthorityBuffer** to pole określa typ jednostki identyfikowanej przez strukturę MQZED wskazanej przez parametr **EntityDataPtr**.

Wartość ta jest jedną z następujących wartości: dla parametru **Filter** wartość MQZAET_NONE jest również poprawna:

MQZAET_PRINCIPAL

Kolumnowo-wierszowa

MQZAET_GROUP

Grupa

Opcje

Typ: MQAUTHOPT-wejście

Opcje. To pole określa opcje, które dają kontrolę nad wyświetlanym profilem. Należy podać jedną z następujących wartości:

MQAUTHOPT_NAME_ALL_MATCHING

Wyświetla wszystkie profile.

MQAUTHOPT_NAME_EXPLICIT

Wyświetla profile o dokładnie takiej samej nazwie, jak nazwa podana w polu **ProfileName**.

Ponadto należy również określić jeden z następujących elementów:

MQAUTHOPT_ENTITY_SET

Wyświetl wszystkie profile, które są używane do obliczenia skumulowanego uprawnienia, które jednostka ma do obiektu określonego przez parametr **ProfileName**. Parametr **ProfileName** nie może zawierać żadnych znaków wieloznacznych.

- Jeśli określony obiekt jest nazwą użytkownika, dla każdego elementu zestawu {entity, groups} zostanie wyświetlony najbardziej odpowiedni profil, który ma zastosowanie do obiektu.
- Jeśli określony obiekt jest grupą, wyświetlany jest najbardziej odpowiedni profil z grupy, która ma zastosowanie do obiektu.
- Jeśli ta wartość zostanie określona, wówczas wartości parametrów **ProfileName**, **ObjectType**, **EntityType** oraz nazwy jednostki określonej w strukturze MQZED produktu **EntityDataPtr** muszą być niepuste.

Jeśli określono wartość MQAUTHOPT_NAME_ALL_MATCHING, można również podać następującą wartość:

MQAUTHOPT_ENTITY_EXPLICIT

Wyświetla profile, które mają dokładnie taką samą nazwę obiektu, jak nazwa jednostki określona w strukturze MQZED produktu **EntityDataPtr**.

Deklaracja C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR48   ProfileName;      /* Profile name */
    MQLONG     ObjectType;       /* Object type */
}
```

```

    MQLONG    Authority;    /* Authority */
    PMQZED    EntityDataPtr; /* Address of MQZED structure identifying an
                             entity */
    MQLONG    EntityType;   /* Entity type */
    MQAUTHOPT Options;      /* Options */
};

```

MQZED-deskryptor jednostki

Struktura MQZED jest używana w wielu wywołaniach usługi autoryzacji w celu określenia jednostki, dla której ma zostać sprawdzona autoryzacja.

Tabela 1. podsumowuje pola w strukturze.

Tabela 209. Pola w MQZED	
Pole	Opis
StrucId	Identyfikator struktury
Wersja	Wersja
EntityNamePtr	Nazwa jednostki
EntityDomainPtr	Wskaźnik domeny jednostki
SecurityId	Identyfikator zabezpieczeń
CorrelationPtr	Wskaźnik korelacji

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZED_STRUC_ID

Identyfikator struktury deskryptora jednostki.

W przypadku języka programowania C zdefiniowana jest również stała MQZED_STRUC_ID_ARRAY; ma taką samą wartość jak MQZED_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQZED_VERSION_1

Struktura deskryptora jednostki Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQZED_CURRENT_VERSION

Bieżąca wersja struktury deskryptora jednostki.

EntityNamePtr

Typ: PMQCHAR-wejście

Nazwa profilu.

Adres nazwy jednostki. Jest to wskaźnik do nazwy obiektu, którego autoryzacja ma zostać sprawdzona.

EntityDomainPtr

Typ: PMQCHAR-wejście

Adres nazwy domeny jednostki. Jest to wskaźnik do nazwy domeny zawierającej definicję obiektu, którego autoryzacja ma zostać sprawdzona.

SecurityId

Typ: MQBYTE40 -dane wejściowe

Uprawnienie.

Identyfikator zabezpieczeń. Jest to identyfikator zabezpieczeń, którego autoryzacja ma zostać sprawdzona.

CorrelationPtr

Typ: MQPTR-wejście

Wskaźnik korelacji. Ułatwia to przekazywanie danych korelacyjnych między funkcją uwierzytelniania użytkownika a innymi odpowiednimi funkcjami OAM.

Deklaracja C

```
typedef struct tagMQZED MQZED;  
struct tagMQZED {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    PMQCHAR    EntityNamePtr;    /* Address of entity name */  
    PMQCHAR    EntityDomainPtr;  /* Address of entity domain name */  
    MQBYTE40   SecurityId;       /* Security identifier */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

Pola

MQZEP-dodawanie punktu wejścia komponentu

Komponent usługi uruchamia tę funkcję podczas inicjowania w celu dodania punktu wejścia do wektora punktu wejścia dla tego komponentu usługi.

Składnia

MQZEP ([*Hconfig*](#) , [*Function*](#) , [*EntryPoint*](#) , [*CompCode*](#) , [*Przyczyna*](#))

Parametry

Konfiguracja Hconfig

Typ: MQHCONFIG-dane wejściowe

Uchwyt konfiguracji. Ten uchwyt reprezentuje komponent, który jest konfigurowany dla tej konkretnej usługi instalowalnej. Musi być ona taka sama, jak komponent przekazany do funkcji konfiguracji komponentu przez menedżer kolejek w wywołaniu inicjowania komponentu.

funkcja

Typ: MQLONG-wejście

Identyfikator funkcji. Poprawne wartości dla tej usługi są definiowane dla każdej instalowalnej usługi.

Jeśli wywołanie MQZEP jest wywoływane więcej niż jeden raz dla tej samej funkcji, to ostatnie wywołanie udostępnia punkt wejścia, który jest używany.

EntryPoint

Typ: PMQFUNC-wejście

Punkt wejścia funkcji. Jest to adres punktu wejścia udostępnianego przez komponent w celu wykonania funkcji.

Wartość NULL jest poprawna i wskazuje, że funkcja nie jest udostępniana przez ten komponent. Zakłada się, że dla punktów wejścia, które nie są zdefiniowane za pomocą MQZEP, przyjmuje się wartość NULL.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') Identyfikator funkcji nie jest poprawny.

BŁĄD MQRC_HCONFIG_ERROR

(2280, X'8E8') Uchwyt konfiguracyjny nie jest poprawny.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

MQZFP-Wolne parametry

Struktura MQZFP jest używana w wywołaniu MQZ_FREE_USER w wywołaniu parametru *FreeParms*. Ten parametr określa dane związane z zasobem, który ma zostać zwolniony.

Tabela 1. podsumowuje pola w strukturze.

Tabela 210. Pola w MQZFP	
Pole	Opis
<u>StrucId</u>	Identyfikator struktury
<u>Wersja</u>	Wersja
<u>Zarezerwowane</u>	Zarezerwowane pole
<u>CorrelationPtr</u>	Wskaźnik korelacji

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZIC_STRUC_ID

Identyfikator struktury kontekstu tożsamości. W przypadku języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY; ta sama wartość ma wartość MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQZFP_VERSION_1

Struktura wolnych parametrów Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQZFP_CURRENT_VERSION

Bieżąca wersja struktury wolnych parametrów.

Zarezerwowane

Typ: MQBYTE8 -dane wejściowe

Zarezerwowane pole. Początkowa wartość jest równa null.

CorrelationPtr

Typ: MQPTR-wejście

Wskaźnik korelacji. Adres danych korelacji odnoszących się do zasobu, który ma zostać zwolniony.

Deklaracja C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQBYTE8     Reserved;        /* Reserved field */
    MQPTR      CorrelationPtr;    /* Address of correlation data */
};
```

Pola

MQZIC-kontekst tożsamości

Struktura MQZIC jest używana w wywołaniu MQZ_AUTHENTICATE_USER w wywołaniu parametru *IdentityContext* .

Struktura MQZIC zawiera informacje o kontekście tożsamości, które identyfikują użytkownika aplikacji, która po raz pierwszy umiała komunikat w kolejce:

- Menedżer kolejek wypełnia pole *UserIdentifier* nazwą identyfikującą użytkownika. Sposób działania menedżera kolejek zależy od środowiska, w którym aplikacja jest uruchomiona.
- Menedżer kolejek wypełnia pole *AccountingToken* znacznikiem lub numerem określonym w aplikacji, w której znajduje się komunikat.
- Aplikacje mogą używać pola *ApplIdentityData* w celu uzyskania dodatkowych informacji, które mają zostać dołączone do użytkownika (na przykład zaszyfrowane hasło).

Odpowiednio autoryzowane aplikacje mogą ustawiać kontekst tożsamości przy użyciu funkcji MQZ_AUTHENTICATE_USER.

Identyfikator zabezpieczeń systemu Windows (SID) jest przechowywany w polu *AccountingToken* , gdy komunikat jest tworzony w ramach produktu IBM MQ for Windows. Identyfikator SID może zostać użyty do uzupełnienia pola *UserIdentifier* i do określenia informacji autoryzacyjnych użytkownika.

Tabela 1. podsumowuje pola w strukturze.

Tabela 211. Pola w MQZIC

Pole	Opis
<u>StrucId</u>	Identyfikator struktury
<u>Wersja</u>	Wersja
<u>UserIdentifier</u>	Identyfikator użytkownika
<u>AccountingToken</u>	Token rozliczania
<u>Dane_tożsamości_aplikacji</u>	Dane tożsamości aplikacji

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZIC_STRUC_ID

Identyfikator struktury kontekstu tożsamości. W przypadku języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY; ta sama wartość ma wartość MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja

Typ: MQLONG-wejście

Numer wersji struktury. Wartość jest następująca:

MQZIC_VERSION_1

Struktura kontekstu tożsamości Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQZIC_CURRENT_VERSION

Bieżąca wersja struktury kontekstu tożsamości.

UserIdentifier

Typ: MQCHAR12 -dane wejściowe

Identyfikator użytkownika. Jest to część kontekstu tożsamości komunikatu. *UserIdentifier* określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Więcej informacji na temat pola *UserIdentifier* zawiera sekcja [“UserIdentifier \(MQCHAR12\)”](#) na stronie 465.

AccountingToken

Typ: MQBYTE32 -dane wejściowe

Token rozliczania. Jest to część kontekstu tożsamości komunikatu. *AccountingToken* umożliwia aplikacji wykonanie pracy wykonanej w wyniku komunikatu, który ma być odpowiednio obciążony. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza jego zawartości. Więcej informacji na temat pola *AccountingToken* zawiera sekcja [“AccountingToken \(MQBYTE32\)”](#) na stronie 420.

Dane_tożsamości_aplikacji

Typ: MQCHAR32 -dane wejściowe

Dane aplikacji odnoszące się do tożsamości. Jest to część kontekstu tożsamości komunikatu. ApplIdentityDane są to informacje, które są definiowane przez pakiet aplikacji, który może być używany do udostępniania dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiona przez aplikacje działające z odpowiednim uprawnieniem użytkownika w celu wskazania, czy dane tożsamości są zaufane. Więcej informacji na temat pola ApplIdentityData zawiera sekcja [“Dane ApplIdentity\(MQCHAR32\)”](#) na stronie 422.

Deklaracja C

```
typedef struct tagMQZED MQZED;  
struct tagMQZED {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQCHAR12   UserIdentifier;    /* User identifier */  
    MQBYTE32   AccountingToken;  /* Accounting token */  
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */  
};
```

Pola

Informacje uzupełniające o interfejsie usług instalowalnych

Te informacje umożliwiają zapoznanie się z informacjami referencyjnymi dla instalowalnych usług.

Dla każdej funkcji znajduje się opis, w tym identyfikator funkcji (dla MQZEP).

Parametry są wyświetlane na liście w kolejności, w jakiej muszą one wystąpić. Wszyscy muszą być obecni.

Po nazwie każdego parametru występuje jego typ danych w nawiasach. Są to elementarne typy danych opisane w sekcji [“Elementarne typy danych”](#) na stronie 1010.

Wywołanie w języku C jest również podane, po opisie parametrów.

MQZEP-dodawanie punktu wejścia komponentu

Ta funkcja jest wywoływana przez komponent usługi, podczas inicjowania, w celu dodania punktu wejścia do wektora punktu wejścia dla tego komponentu usługi.

Składnia

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

Parametry

Wywołanie MQZEP ma następujące parametry.

Hconfig (MQHCONFIG)-dane wejściowe

Uchwyt konfiguracji.

Ten uchwyt reprezentuje komponent, który jest konfigurowany dla tej konkretnej usługi instalowalnej. Musi być ona taka sama, jak ta przekazana do funkcji konfiguracji komponentu przez menedżer kolejek w wywołaniu inicjowania komponentu.

Funkcja (MQLONG)-dane wejściowe

Identyfikator funkcji.

Poprawne wartości dla tej usługi są definiowane dla każdej instalowalnej usługi. Jeśli program MQZEP jest wywoływany więcej niż jeden raz dla tej samej funkcji, to ostatnie wywołanie udostępnia punkt wejścia, który jest używany.

EntryPoint (PMQFUNC)-dane wejściowe

Punkt wejścia funkcji.

Jest to adres punktu wejścia udostępnianego przez komponent w celu wykonania funkcji. Wartość NULL jest poprawna i wskazuje, że funkcja nie jest udostępniana przez ten komponent. Dla punktów wejścia, które nie są zdefiniowane za pomocą MQZEP, przyjmowana jest wartość NULL .

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') Identyfikator funkcji nie jest poprawny.

BŁĄD MQRC_HCONFIG_ERROR

(2280, X'8E8') Uchwyt konfiguracyjny nie jest poprawny.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

MQHCONFIG-uchwyt konfiguracji

Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla określonej usługi instalowalnej. Uchwyt konfiguracji musi być wyrównany względem jego naturalnej granicy.

Aplikacje muszą testować zmienne tego typu tylko w celu zapewnienia równości.

Deklaracja C

```
typedef void MQPOINTER MQHCONFIG;
```

PMQFUNC-wskaźnik do funkcji

Wskaźnik do funkcji.

Deklaracja C

```
typedef void MQPOINTER PMQFUNC;
```


MQZ_AUTHENTICATE_USER-Uwierzytelnienie użytkownika

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 . Jest ona wywoływana przez menedżer kolejek w celu uwierzytelnienia użytkownika lub ustawiania pól kontekstu tożsamości.

Jest ona wywoływana, gdy zostanie utworzony kontekst aplikacji użytkownika produktu IBM MQ . Taka sytuacja ma miejsce podczas wywołań połączenia w punkcie, w którym zainicjowano kontekst użytkownika aplikacji, a także w każdym punkcie, w którym zmieniono kontekst użytkownika aplikacji. Za każdym razem, gdy nawiąże się połączenie, informacje o kontekście użytkownika aplikacji są ponownie uzyskiwane w polu *IdentityContext* .

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_AUTHENTICATE_USER.

Składnia

MQZ_AUTHENTICATE_USER (*QMgrName*, *SecurityParms*, *ApplicationContext*, *IdentityContext*, *CorrelationPtr*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parametry

Wywołanie MQZ_AUTHENTICATE_USER ma następujące parametry.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero. Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

SecurityParms (MQCSP)-dane wejściowe

Parametry zabezpieczeń.

Dane odnoszące się do identyfikatora użytkownika, hasła i typu uwierzytelniania.

Podczas wywołania MQCONN MQI ten parametr zawiera wartości NULL lub wartości domyślne.

ApplicationContext (MQZAC)-dane wejściowe

Kontekst aplikacji.

Dane odnoszące się do aplikacji wywołującej. Szczegółowe informacje na ten temat zawiera sekcja “MQZAC-kontekst aplikacji” na stronie 1811. Podczas wywoływania wszystkich wywołań MQI MQCONN lub MQCONNX informacje o kontekście użytkownika w strukturze MQZAC są ponownie nabywane.

IdentityContext (MQZIC)-wejście/wyjście

Kontekst tożsamości.

W przypadku danych wejściowych dla funkcji uwierzytelniania użytkownika identyfikuje bieżący kontekst tożsamości. Funkcja uwierzytelniania użytkownika może to zmienić, co oznacza, że menedżer kolejek adoptuje nowy kontekst tożsamości. Więcej informacji na temat struktury MQZIC zawiera sekcja “MQZIC-kontekst tożsamości” na stronie 1818 .

CorrelationPtr (MQPTR)-dane wyjściowe

Wskaźnik korelacji.

Określa adres wszystkich danych korelacji. Wskaźnik ten jest następnie przekazywany do innych wywołań OAM.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji komponentów. Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Flaga kontynuacji.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od innych komponentów.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,  
                        IdentityContext, &CorrelationPtr, ComponentData,  
                        &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCSP     SecurityParms;      /* Security parameters */  
MQZAC     ApplicationContext; /* Application context */  
MQZIC     IdentityContext;    /* Identity context */  
MQPTR     CorrelationPtr;     /* Correlation pointer */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY-sprawdzanie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonywania określonego działania lub działań na określonym obiekcie.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_CHECK_AUTHORITY.

Składnia

MQZ_CHECK_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

Parametry

Wywołanie MQZ_CHECK_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero. Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe

Nazwa jednostki.

Nazwa obiektu, którego autoryzacja do obiektu ma zostać sprawdzona. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Nie jest istotne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest ona znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (do której należą wszystkie jednostki). Pusta nazwa jest poprawna i może być używana w ten sposób.

EntityType (MQLONG)-dane wejściowe

Typ jednostki.

Typ jednostki określony przez produkt *EntityName*. Jest to jedna z poniższych nazw:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ jednostki określony przez produkt *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie (MQLONG)-dane wejściowe

Uprawnienie do sprawdzenia.

Jeśli sprawdzana jest jedna autoryzacja, to pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli sprawdzana jest więcej niż jedna autoryzacja, to jest to bitowe LUB odpowiadające im stałe MQZAO_*.

Do korzystania z wywołań MQI stosowane są następujące autoryzacje:

MQZAO_CONNECT

Możliwość korzystania z wywołania MQCONN.

MQZAO_PRZEGLĄDANIE

Możliwość korzystania z wywołania MQGET z opcją przeglądania.

Pozwala to na określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT

Możliwość korzystania z wywołania MQGET z opcją wejściową.

Umożliwia to określenie w wywołaniu MQOPEN opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_Q_Q_Q_DEF.

MQZAO_OUTPUT

Możliwość korzystania z wywołania MQPUT.

Pozwala to na określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_ZAPYTANIE_O

Możliwość korzystania z wywołania MQINQ.

Pozwala to na określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET

Możliwość korzystania z wywołania MQSET.

Pozwala to na określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT

Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_PASS_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1.

MQZAO_PASS_ALL_CONTEXT

Możliwość przekazania całego kontekstu.

Pozwala to na określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT

Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcję MQPMO_SET_IDENTITY_CONTEXT, która ma zostać określona w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_ALL_CONTEXT,

Możliwość ustawienia całego kontekstu.

Pozwala to na określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY

Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN oraz opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu komendy MQPUT1 .

MQZAO_ALL_MQI

Wszystkie autoryzacje MQI.

Umożliwia to wszystkie opisane wcześniej autoryzacje.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE

Możliwość tworzenia obiektów o określonym typie.

MQZAO_DELETE

Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY

Możliwość wyświetlania atrybutów określonego obiektu.

ZMIANA MQZAO_CHANGE

Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR

Możliwość usuwania wszystkich komunikatów z określonej kolejki.

MQZAO_AUTORYZACJA

Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL

Możliwość uruchamiania, zatrzymywania lub wysyłania pakietów ping do obiektu kanału innego niż klienta.

MQZAO_CONTROL_EXTENDED

Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwej wiadomości na obiekcie kanału innego niż klient.

MQZAO_ALL_ADMIN

Wszystkie autoryzacje administracyjne, inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do korzystania z interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_ALL

Wszystkie autoryzacje, inne niż MQZAO_CREATE.

MQZAO_NONE

Brak autoryzacji.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to taki sam efekt jak MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptažania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                    ObjectType, Authority, ComponentData,  
                    &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */
```

MQLONG	Authority;	/* Authority to be checked */
MQBYTE	ComponentData[n];	/* Component data */
MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_CHECK_PRIVILEGED-sprawdź, czy użytkownik jest uprzywilejowany

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_6 i jest wywoływana przez menedżer kolejek w celu określenia, czy określony użytkownik jest uprzywilejowanym użytkownikiem.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_CHECK_PRIVILEGED.

Składnia

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

EntityData

Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do podmiotu, który ma zostać sprawdzony. Więcej informacji na ten temat zawiera sekcja [“MQZED-deskryptor jednostki”](#) na stronie 1774.

EntityType

Typ: MQLONG-wejście

Typ jednostki. Typ obiektu określony przez obiekt EntityData. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ComponentData

Typ: MQBYTE ×ComponentDataLength-input/output

Dane komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Indykator kontynuacji ustawiony przez komponent. Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_CHECK_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

Jeśli wywołanie do komponentu nie powiedzie się (to jest, *CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* to MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek będzie nadal wywoływać inne komponenty, jeśli są jakieś.

Jeśli wywołanie powiedzie się (tj. *CompCode* zwraca wartość MQCC_OK), żadne inne komponenty nie są wywoływane bez względu na to, jakie jest ustawienie *Kontynuacja*.

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie są wywoływane żadne inne komponenty, a błąd jest zwracany do menedżera kolejek. Komponenty nie mają wiedzy o poprzednich wywołaniach, dlatego parametr *Continuation* jest zawsze ustawiony na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

reason

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_PRIVILEGED

(2584, X'A18') Ten użytkownik nie jest uprzywilejowanym identyfikatorem użytkownika.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Kody zakończenia i kody przyczyny interfejsu API](#).

Wywołanie C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                     ComponentData, &Continuation,
                     &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
```


MQLONG	EntityType;	/* Entity type */
MQBYTE	ComponentData[n];	/* Component data */
MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_COPY_ALL_AUTHORITY-kopiowanie wszystkich uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji. Jest on wywoływany przez menedżer kolejek w celu skopiowania wszystkich uprawnień, które aktualnie są w mocy, dla obiektu odwołania do innego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_COPY_ALL_AUTHORITY.

Składnia

MQZ_COPY_ALL_AUTHORITY (*QMgrName, RefObjectName, ObjectName, ObjectType, ComponentData, Continuation, CompCode, Reason*)

Parametry

Wywołanie MQZ_COPY_ALL_AUTHORITY ma następujące parametry:

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

RefObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu odniesienia.

Nazwa obiektu odniesienia, autoryzacje, dla których mają być skopiowane. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać ustawione dostępy. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ obiektu określony przez *RefObjectName* i *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE**ComponentData (MQBYTE x ComponentDataLength)-input/output**

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku MQZ_COPY_ALL_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') Nieznany obiekt referencyjny.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,  
                        ComponentData, &Continuation, &CompCode,  
                        &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  RefObjectName;      /* Reference object name */  
MQCHAR48  ObjectName;        /* Object name */  
MQLONG    ObjectType;        /* Object type */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY-uprawnienie do usuwania

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek w celu usunięcia wszystkich autoryzacji powiązanych z określonym obiektem.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_AUTHORITY.

Składnia

MQZ_DELETE_AUTHORITY (*QMgrName*, *ObjectName*, *ObjectType*,
ComponentData, *Continuation*, *CompCode*, *Reason*)

Parametry

Wywołanie MQZ_DELETE_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać usunięte dostępy. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ jednostki określony przez produkt *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nastuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE**ComponentData (MQBYTE x ComponentDataLength)-input/output**

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_DELETE_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,  
                      &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA-Dane o uprawnieniach Enumerate

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_4 i jest wywoływana wielokrotnie przez menedżer kolejek w celu pobrania wszystkich danych uprawnień, które są zgodne z kryteriami wyboru określonymi podczas pierwszego wywołania.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_ENUMERATE_AUTHORITY_DATA.

Składnia

MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_ENUMERATE_AUTHORITY_DATA ma następujące parametry.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

StartEnumeration (MQLONG)-dane wejściowe

Flaga wskazująca, czy wywołanie powinno rozpoczynać wyliczenie.

Wskazuje, czy wywołanie powinno zaczynać się od wyliczania danych uprawnień, czy kontynuować wyliczanie danych uprawnień rozpoczętych przez poprzednie wywołanie MQZ_ENUMERATE_AUTHORITY_DATA. Wartość ta jest jedną z następujących wartości:

MQZSE_START

Początkowe wyliczenie.

Wywołanie jest wywoływane z tą wartością, aby rozpocząć wyliczanie danych uprawnień.

Parametr *Filter* określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez te i kolejne wywołania.

MQZSE_CONTINUE

Kontynuuj wyliczanie.

Wywołanie jest wywoływane z tą wartością, aby kontynuować wyliczanie danych uprawnień. Parametr *Filter* jest w tym przypadku ignorowany i może zostać określony jako wskaźnik pusty (kryteria wyboru są określone przez parametr *Filter* określony przez wywołanie, które miało *StartEnumeration* ustawione na wartość MQZSE_START).

Filtr (MQZAD)-dane wejściowe

Filtr.

Jeśli parametr *StartEnumeration* ma wartość MQZSE_START, *Filter* określa kryteria wyboru, które mają być używane do wybierania danych uprawnień do zwrotu. Jeśli *Filter* jest wskaźnikiem zerowym, nie są używane żadne kryteria wyboru, to znaczy, że zwracane są wszystkie dane uprawnień. Szczegółowe informacje na temat kryteriów wyboru, które można wykorzystać, zawiera sekcja “MQZAD (dane uprawnień) w systemie IBM i” na stronie 1813 .

Jeśli parametr *StartEnumeration* ma wartość MQZSE_CONTINUE, *Filter* jest ignorowany i może zostać określony jako wskaźnik pusty.

AuthorityBufferLength (MQLONG)-dane wejściowe

Długość *AuthorityBuffer*.

Jest to długość w bajtach parametru *AuthorityBuffer* . Bufor uprawnień musi być wystarczająco duży, aby pomieścić dane, które mają zostać zwrócone.

AuthorityBuffer (MQZAD)-dane wyjściowe

Dane uprawnień.

Jest to bufor, w którym zwracane są dane uprawnień. Bufor musi być wystarczająco duży, aby pomieścić strukturę MQZAD, strukturę MQZED oraz najdłuższą zdefiniowaną nazwę jednostki i najdłuższą zdefiniowaną nazwę domeny.

Uwaga: Ten parametr jest zdefiniowany jako zmaterializowana tabela zapytania (MQZAD), ponieważ wartość MQZAD zawsze występuje na początku buforu. Jeśli jednak bufor jest w rzeczywistości zadeklarowany jako zmaterializowana tabela zapytania (MQZAD), bufor będzie zbyt mały-musi być większy niż zmaterializowana tabela zapytania (MQZAD), dzięki czemu może pomieścić nazwy obiektów MQZAD, MQZED oraz jednostki i domeny.

AuthorityDataLength (MQLONG)-dane wyjściowe

Długość danych zwracanych w produkcie *AuthorityBuffer*.

Jest to długość danych zwracanych w produkcie *AuthorityBuffer*. Jeśli bufor uprawnień jest zbyt mały, parametr *AuthorityDataLength* jest ustawiony na długość wymaganego buforu, a wywołanie zwraca kod zakończenia MQCC_FAILED i kod przyczyny MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku komendy MQZ_ENUMERATE_AUTHORITY_DATA ma to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Parametr długości buforu nie jest poprawny.

MQRC_NO_DATA_AVAILABLE

(2379, X'94B') Brak dostępnych danych.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,  
                               AuthorityBufferLength,  
                               &AuthorityBuffer,  
                               &AuthorityDataLength, ComponentData,  
                               &Continuation, &CompCode,  
                               &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

MQCHAR48	QMgrName;	/* Queue manager name */
MQLONG	StartEnumeration;	/* Flag indicating whether call should start enumeration */
MQZAD	Filter;	/* Filter */
MQLONG	AuthorityBufferLength;	/* Length of AuthorityBuffer */
MQZAD	AuthorityBuffer;	/* Authority data */
MQLONG	AuthorityDataLength;	/* Length of data returned in AuthorityBuffer */
MQBYTE	ComponentData[n];	/* Component data */
MQLONG	Continuation;	/* Continuation indicator set by component */
MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_FREE_USER-użytkownik wolny

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu zwolnienia powiązanego z nim przydzielonego zasobu. Jest on wywoływany, gdy aplikacja zakończyła działanie we wszystkich kontekstach użytkownika, na przykład podczas wywołania MQI MQDISC.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_FREE_USER.

MQZ_GET_AUTHORITY-pobieranie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu pobrania uprawnień, które jednostka musi uzyskać w celu uzyskania dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_AUTHORITY.

Składnia

MQZ_GET_AUTHORITY (*QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

Parametry

Wywołanie MQZ_GET_AUTHORITY ma następujące parametry:

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe

Nazwa jednostki.

Nazwa jednostki, której dostęp do obiektu ma zostać pobrany. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe

Typ jednostki.

Typ jednostki określony przez produkt *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, dla którego ma zostać pobrany organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ jednostki określony przez produkt *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie (MQLONG)-dane wyjściowe

Organ jednostki.

Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_*.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku MQZ_GET_AUTHORITY ma to ten sam efekt, co komenda MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_GET_AUTHORITY (QMGrName, EntityName, EntityType, ObjectName,  
                  ObjectType, &Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMGrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority of entity */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */  
MQLONG    CompCode;           /* Completion code */  
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY-Uzyskanie jawnego uprawnienia

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu pobrania uprawnień, które ma dostęp do określonego obiektu przez nazwaną grupę (ale bez dodatkowego uprawnienia grupy **nobody**), lub uprawnienia, które grupa podstawowa nazwanej nazwy użytkownika ma do dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_GET_EXPLICIT_AUTHORITY.

Składnia

MQZ_GET_EXPLICIT_AUTHORITY (*QMGrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason*)

Parametry

Wywołanie MQZ_GET_EXPLICIT_AUTHORITY ma następujące parametry.

QMGrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe

Nazwa jednostki.

Nazwa jednostki, z której ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe

Typ jednostki.

Typ jednostki określony przez produkt *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, dla którego ma zostać pobrany organ jednostki. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ jednostki określony przez produkt *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwanie.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie (MQLONG)-dane wyjściowe

Organ jednostki.

Jeśli jednostka ma jedno uprawnienie, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli ma więcej niż jeden uprawnienie, pole to jest bitowe OR dla odpowiednich stałych MQZAO_*.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indyktor kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku MQZ_GET_EXPLICIT_AUTHORITY ma to taki sam efekt jak MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptażania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,  
                             ObjectName, ObjectType, &Authority,  
                             ComponentData, &Continuation,  
                             &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCHAR12  EntityName;         /* Entity name */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                               component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY-inicjowanie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się, że wywołanie MQZEP będzie możliwe w celu udostępnienia informacji do menedżera kolejek.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_INIT_AUTHORITY.

Składnia

MQZ_INIT_AUTHORITY (*Hconfig, Options, QMgrName, ComponentDataLength, ComponentData, Version, CompCode, Reason*)

Parametry

Wywołanie MQZ_INIT_AUTHORITY ma następujące parametry.

Hconfig (MQHCONFIG)-dane wejściowe

Uchwyt konfiguracji.

Ten uchwyt reprezentuje określony komponent, który jest inicjowany. Ma ona być używana przez komponent podczas wywoływania menedżera kolejek przy użyciu funkcji MQZEP.

Opcje (MQLONG)-dane wejściowe

Opcje inicjowania.

Jest to jedna z poniższych nazw:

MQZIO_PRIMARY

Inicjowanie podstawowe.

MQZIO_SECONDARY

Inicjowanie wtórne.

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

ComponentDataLength (MQLONG)-dane wejściowe

Długość danych komponentu.

Długość w bajtach obszaru *ComponentData*. Ta długość jest zdefiniowana w danych konfiguracji komponentu.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Jest on inicjowany dla wszystkich zer przed wywołaniem podstawowej funkcji inicjowania komponentu. Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji (w tym funkcję inicjowania) są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Wersja (MQLONG)-wejście/wyjście

Numer wersji.

W przypadku wejścia do funkcji inicjowania identyfikuje on *najwyższy* numer wersji obsługiwany przez menedżer kolejek. Funkcja inicjowania musi to zmienić, jeśli jest to konieczne, do wersji interfejsu, który jest obsługiwany przez *it*. Jeśli po powrocie menedżer kolejek nie obsługuje wersji zwracanej przez komponent, wywołuje funkcję MQZ_TERM_AUTHORITY komponentu i nie korzysta z tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1

Wersja 1.

MQZAS_VERSION_2

Wersja 2.

MQZAS_VERSION_3

Wersja 3.

MQZAS_VERSION_4

Wersja 4.

MQZAS_VERSION_5

Wersja 5.

MQZAS_VERSION_6

Wersja 6.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanego powodu.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptažania.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
```

```
ComponentData, &Version, &CompCode,  
&Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Initialization options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQLONG     ComponentDataLength; /* Length of component data */  
MQBYTE     ComponentData[n];  /* Component data */  
MQLONG     Version;           /* Version number */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_INQUIRE-zapytanie o usługę autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu wysłania zapytania o obsługiwane funkcje. W przypadku użycia wielu komponentów usług komponenty usług są wywoływane w kolejności odwrotnej do kolejności, w jakiej zostały zainstalowane.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INQUIRE.

Składnia

MQZ_INQUIRE

(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_INQUIRE ma następujące parametry:

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

SelectorCount (MQLONG)-dane wejściowe

Liczba selektorów.

Liczba selektorów podanych w parametrze Selektory.

Wartość musi być z zakresu od zera do 256.

Selektory (MQLONG x SelectorCount)-dane wejściowe

Selektory.

Tablica selektorów. Każdy selektor identyfikuje wymagany atrybut i musi mieć jeden z następujących typów:

- MQIACF_ * (liczba całkowita)
- MQCACF_ * (znak)

Selektory mogą być określane w dowolnej kolejności. Liczba selektorów w tablicy jest wskazywana przez parametr SelectorCount .

Atrybuty całkowitoliczbowe zidentyfikowane przez selektory są zwracane w parametrze IntAttrs w tej samej kolejności, w jakiej są wyświetlane w selektorach.

Atrybuty znakowe identyfikowane przez selektory są zwracane w parametrze CharAttrs w takiej samej kolejności, w jakiej są widoczne w selektorach.

IntAttrCount (MQLONG)-dane wejściowe

Liczba atrybutów całkowitych.

Liczba atrybutów całkowitoliczbowych podanych w parametrze IntAttr.

Wartość musi być z zakresu od 0 do 256.

IntAttr (liczba operacji MQLONG x IntAttrCount)-dane wyjściowe

Atrybuty całkowite.

Tablica atrybutów całkowitoliczbowych. Atrybuty całkowitoliczbowe są zwracane w tej samej kolejności, w jakiej znajdują się odpowiednie selektory całkowite w tablicy Selektory.

CharAttrCount (MQLONG)-dane wejściowe

Długość buforu atrybutów znaków.

Długość (w bajtach) parametru CharAttr (CharAttr).

Wartość musi co najmniej być sumą długości żądanych atrybutów znakowych. Jeśli atrybuty znaków nie są wymagane, wartość zero jest poprawną wartością.

CharAttr (liczba tabel MQLONG x CharAttr)-dane wyjściowe

Bufor atrybutów znaków.

Bufor zawierający atrybuty znaków, konkatelowany razem. Atrybuty znakowe są zwracane w takiej samej kolejności, w jakiej znajdują się odpowiednie selektory znaków w tablicy Selektory.

Długość buforu jest nadawana przez parametr CharAttrCount.

SelectorReturned (liczba operacji MQLONG x Selector)-dane wejściowe

Selektor został zwrócony.

Tablica wartości identyfikujących, które atrybuty zostały zwrócone z zestawu żądanych przez selektory w parametrze Selektory. Liczba wartości w tej tablicy jest wskazywana przez parametr SelectorCount. Każda wartość w tablicy odnosi się do selektora z odpowiedniej pozycji w tablicy Selektory. Każda wartość jest jedną z następujących wartości:

MQZSL_RETURNED

Atrybut żądany przez odpowiedni selektor w parametrze Selektory został zwrócony.

MQZSL_NOT_RETURNED

Atrybut żądany przez odpowiedni selektor w parametrze Selektory nie został zwrócony.

Tablica jest inicjowana ze wszystkimi wartościami jako *MQZSL_NOT_RETURNED*. Gdy komponent usługi autoryzacji zwraca atrybut, ustawia odpowiednią wartość w tablicy na wartość *MQZSL_RETURNED*. Umożliwia to innym komponentom usług autoryzacji, do których jest nawiązywać zapytanie, identyfikowanie atrybutów, które zostały już zwrócone.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Flaga kontynuacji.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od innych komponentów.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_WARNING,

Zakończenie częściowe.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli *CompCode* to MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

Za mało miejsca dla atrybutów znakowych.

MQRC_INT_COUNT_TOO_SMALL

Zbyt mało miejsca dla atrybutów całkowitych.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR,

Liczba selektorów jest niepoprawna.

MQRC_SELECTOR_ERROR,

Selektor atrybutu jest niepoprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED

Określono zbyt wiele selektorów.

MQRC_INT_ATTR_COUNT_ERROR

Liczba atrybutów całkowitych nie jest poprawna.

MQRC_INT_ATTRS_ARRAY_ERROR

Tablica atrybutów liczb całkowitych nie jest poprawna.

MQRC_CHAR_ATTR_LENGTH_ERROR

Liczba atrybutów znakowych nie jest poprawna.

MQRC_CHAR_ATTRS_ERROR

Łańcuch atrybutów znakowych nie jest poprawny.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,  
              &IntAttrs, CharAttrLength, &CharAttrs,  
              SelectorReturned, ComponentData, &Continuation,  
              &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQLONG    SelectorCount;      /* Selector count */  
MQLONG    Selectors[n];       /* Selectors */  
MQLONG    IntAttrCount;       /* IntAttrs count */
```

MQLONG	IntAttrs[n];	/* Integer attributes */
MQLONG	CharAttrCount;	/* CharAttrs count */
MQLONG	CharAttrs[n];	/* 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-Odśwież wszystkie autoryzacje

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_3 . Jest ona wywoływana przez menedżer kolejek w celu odświeżenia listy autoryzacji przechowywanych wewnętrznie przez komponent.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_REFRESH_CACHE (8L).

Składnia

MQZ_REFRESH_CACHE

(QMgrName, ComponentData, Kontynuacja, CompCode, Przyczyna)

Parametry

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał go w żaden zdefiniowany sposób.

ComponentData (MQBYTE x ComponentDataLength) -input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Wszelkie zmiany wprowadzone w nim przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuowanie (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku MQZ_REFRESH_CACHE jest to ten sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli parametr *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQBYTE    ComponentData[n];  /* Component data */  
MQLONG    Continuation;      /* Continuation indicator set by  
                             component */  
MQLONG    CompCode;          /* Completion code */  
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY-uprawnienie do ustawiania

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu ustawienia uprawnień, które jednostka musi uzyskać w celu uzyskania dostępu do określonego obiektu.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja przestania wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić przy użyciu tej funkcji.

Składnia

MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_SET_AUTHORITY ma następujące parametry:

QMgrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe

Nazwa jednostki.

Nazwa obiektu, dla którego ma zostać ustawiony dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe

Typ jednostki.

Typ jednostki określony przez produkt *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

ObjectName (MQCHAR48)-dane wejściowe

Nazwa obiektu.

Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków. Jeśli jest ona krótsza niż jest dopełniona z prawej strony odstępami. Nazwa nie została zakończona znakiem o kodzie zero.

Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, nazwa ta jest taka sama jak nazwa *QMgrName*.

ObjectType (MQLONG)-dane wejściowe

Typ obiektu.

Typ jednostki określony przez produkt *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie (MQLONG)-dane wejściowe

Uprawnienie do sprawdzenia.

Jeśli ustawiona jest jedna autoryzacja, pole to jest równe odpowiedniej operacji autoryzacji (stała MQZAO_*). Jeśli ustawiona jest więcej niż jedna autoryzacja, to jest to bitowe OR odpowiednich stałych MQZAO_*.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe

Indykator kontynuacji ustawiony przez komponent.

Można określić następujące wartości:

MQZCI_DEFAULT

Kontynuacja zależna od menedżera kolejek.

W przypadku MQZ_SET_AUTHORITY ma to ten sam efekt co MQZCI_STOP.

MQZCI_CONTINUE

Przejdź do następnego komponentu.

MQZCI_STOP

Nie należy kontynuować z następnym komponentem.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Brak uprawnień do dostępu.

MQRC_SERVICE_ERROR,

(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedoptažania.

MQRC_UNKNOWN_ENTITY,

(2292, X'8F4') Obiekt nieznany do obsługi.

Wywołanie C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,  
                  ObjectType, Authority, ComponentData,  
                  &Continuation, &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */  
MQCHAR12  EntityName;         /* Entity name */  
MQLONG    EntityType;         /* Entity type */  
MQCHAR48  ObjectName;         /* Object name */  
MQLONG    ObjectType;         /* Object type */  
MQLONG    Authority;          /* Authority to be checked */  
MQBYTE    ComponentData[n];   /* Component data */  
MQLONG    Continuation;       /* Continuation indicator set by  
                               component */
```

MQLONG	CompCode;	/* Completion code */
MQLONG	Reason;	/* Reason code qualifying CompCode */

MQZ_TERM_AUTHORITY-kończenie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek, gdy nie wymaga ona już usług tego komponentu. Funkcja musi wykonać procedurę czyszczącą wymagającą przez komponent.

Identyfikatorem funkcji dla tej funkcji (dla MQZEP) jest MQZID_TERM_AUTHORITY.

Składnia

MQZ_TERM_AUTHORITY (*Hconfig, Options, QMgrName, ComponentData, CompCode, Reason*)

Parametry

Wywołanie MQZ_TERM_AUTHORITY ma następujące parametry:

Hconfig (MQHCONFIG)-dane wejściowe

Uchwyt konfiguracji.

Ten uchwyt reprezentuje określony komponent, który został zakończony.

Opcje (MQLONG)-dane wejściowe

Opcje zakończenia.

Jest to jedna z poniższych nazw:

MQZTO_PRIMARY

Zakończenie podstawowe.

MQZTO_SECONDARY

Zakończenie wtórne.

QMGrName (MQCHAR48)-dane wejściowe

Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Ta nazwa jest dopełniona spacjami do pełnej długości parametru; nazwa nie została zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent był używany w żaden zdefiniowany sposób.

ComponentData (MQBYTE x ComponentDataLength)-input/output

Dane komponentu.

Dane te są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu; wszystkie zmiany wprowadzone w tym komponentcie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane po następnym wywołaniu funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżera kolejek w parametrze *ComponentDataLength* w wywołaniu MQZ_INIT_AUTHORITY.

Po zakończeniu wywołania MQZ_TERM_AUTHORITY menedżer kolejek odrzuci te dane.

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK

Zakończenie powiodło się.

MQCC_FAILED

Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny kwalifikujący *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE

(0, X'000 ') Nie ma powodu do zgłoszenia.

Jeśli parametr *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Niedostępna usługa niedopłażania.

MQRC_TERMINATION_FAILED

(2287, X'8FF') Wygaśnienie nie powiodło się z niezdefiniowanego powodu.

Więcej informacji na temat tych kodów przyczyny można znaleźć w sekcji [Komunikaty i kody przyczyny](#).

Wywołanie C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,  
                    &CompCode, &Reason);
```

Parametry przekazane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;           /* Configuration handle */  
MQLONG     Options;           /* Termination options */  
MQCHAR48   QMgrName;         /* Queue manager name */  
MQBYTE     ComponentData[n]; /* Component data */  
MQLONG     CompCode;          /* Completion code */  
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC-kontekst aplikacji

Ten parametr określa dane związane z aplikacją wywołującą.

Struktura MQZAC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru *ApplicationContext*.

Pola

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość jest następująca:

MQZAC_STRUC_ID

Identyfikator struktury kontekstu aplikacji.

Dla języka programowania C jest również zdefiniowana stała zmienna MQZAC_STRUC_ID_ARRAY; ma taką samą wartość jak MQZAC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe dla usługi.

Wersja (MQLONG)

Numer wersji struktury.

Wartość jest następująca:

MQZAC_VERSION_1

Struktura kontekstu aplikacji Version-1.

Następująca stała określa numer wersji bieżącej wersji:

MQZAC_CURRENT_VERSION

Bieżąca wersja struktury kontekstu aplikacji.

To jest pole wejściowe dla usługi.

ProcessId (MQPID)

Identyfikator procesu.

Identyfikator procesu aplikacji.

ThreadId (MQTID)

Identyfikator wątku.

Identyfikator wątku aplikacji.

ApplName (MQCHAR28)

Nazwa aplikacji.

Nazwa aplikacji.

UserID (MQCHAR12)

Identyfikator użytkownika.

W systemach IBM i profil użytkownika, w którym zostało utworzone zadanie aplikacji. (IBM i: po zakończeniu wymiany profilu za pomocą funkcji API QWTSETP w zadaniu aplikacji zwracany jest bieżący profil użytkownika).

Identyfikator EffectiveUser(MQCHAR12)

Efektywny identyfikator użytkownika.

W systemie IBM i jest to profil bieżącego użytkownika zadania aplikacji.

Środowisko (MQLONG)

Środowisko.

To pole określa środowisko, z którego połączenie zostało wykonane.

Może to mieć jedną z następujących wartości:

MQXE_COMMAND_SERVER

Serwer komend.

MQXE_MQSC

Interpreter komend runmqsc .

MQXE_MCA

Agent kanału komunikatów

MQXE_INNY

Niezdefiniowane środowisko

CallerType (MQLONG)

Typ programu wywołującego.

To pole określa typ programu, który wywołał wywołanie.

Może to mieć jedną z następujących wartości:

MQXACT_EXTERNAL

Wywołanie jest zewnętrzne w stosunku do menedżera kolejek.

MQXACT_INTERNAL

Wywołanie jest wewnętrzne dla menedżera kolejek.

AuthenticationType (MQLONG)

Typ uwierzytelniania.

To pole określa typ wykonywanego uwierzytelniania.

Może to mieć jedną z następujących wartości:

MQZAT_INITIAL_CONTEXT

Wywołanie uwierzytelniania jest spowodowane zainicjowaniem kontekstu użytkownika. Ta wartość jest używana podczas wywołania funkcji MQCONN lub MQCONNX .

MQZAT_CHANGE_CONTEXT,

Wywołanie uwierzytelniania jest spowodowane zmianą kontekstu użytkownika. Ta wartość jest używana, gdy agent MCA zmienia kontekst użytkownika.

v

BindType (MQLONG)

Typ powiązania.

To pole określa typ powiązania, który ma być używany.

Może to mieć jedną z następujących wartości:

MQCNO_FASTPATH_BINDING

Powiązanie krótkiej ścieżki.

MQCNO_SHARED_BINDING

Powiązanie współużytkowane.

MQCNO_ISOLATED_BINDING

Powiązanie izolowane.

Deklaracja C

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQPID      ProcessId;        /* Process identifier */
    MQTID      ThreadId;         /* Thread identifier */
    MQCHAR28    ApplName;        /* Application name */
    MQCHAR12    UserID;          /* User identifier */
    MQCHAR12    EffectiveUserID; /* Effective user identifier */
    MQLONG     Environment;      /* Environment */
    MQLONG     CallerType;       /* Caller type */
    MQLONG     AuthenticationType; /* Authentication type */
    MQLONG     BindType;         /* Bind type */
};
```

MQZAD (dane uprawnień) w systemie IBM i

Struktura MQZAD jest używana w wywołaniu MQZ_ENUMERATE_AUTHORITY_DATA dla dwóch parametrów.

Dostępne są następujące dwa parametry:

- Parametr MQZAD jest używany dla parametru **Filter**, który jest wprowadzany do wywołania. Ten parametr określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez wywołanie.
- Produkt MQZAD jest również używany w przypadku parametru **AuthorityBuffer**, który jest wyjściem wywołania. Ten parametr określa autoryzacje dla jednej kombinacji nazwy profilu, typu obiektu i obiektu.

Pola

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość jest następująca:

MQZAD_STRUC_ID

Identyfikator struktury danych uprawnień.

Dla języka programowania C zdefiniowana jest również stała MQZAD_STRUC_ID_ARRAY; ma taką samą wartość jak MQZAD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe dla usługi.

Wersja (MQLONG)

Numer wersji struktury.

Wartość jest następująca:

MQZAD_VERSION_1

Struktura danych uprawnień Version-1 .

Następująca stała określa numer wersji bieżącej wersji:

MQZAD_CURRENT_VERSION

Bieżąca wersja struktury danych uprawnień.

To jest pole wejściowe dla usługi.

ProfileName (MQCHAR48)

Nazwa profilu.

W przypadku parametru **Filter** to pole jest nazwą profilu, z którego wymagane są dane uprawnień. Jeśli nazwa jest całkowicie pusta aż do końca pola lub pierwszego znaku o kodzie zero, zwracane są dane o uprawnieniach dla wszystkich nazw profili.

W przypadku parametru **AuthorityBuffer** to pole jest nazwą profilu, który jest zgodny z podanymi kryteriami wyboru.

ObjectType (MQLONG)

Typ obiektu.

W przypadku parametru **Filter** to pole jest typem obiektu, dla którego wymagane są dane uprawnień. Jeśli wartością jest MQOT_ALL, zwracane są dane o uprawnieniach dla wszystkich typów obiektów.

W przypadku parametru **AuthorityBuffer** to pole jest typem obiektu, do którego ma zastosowanie profil identyfikowany przez produkt **ProfileName** .

Wartość ta jest jedną z następujących wartości: dla parametru **Filter** wartość MQOT_ALL jest również poprawna:

MQOT_AUTH_INFO

Informacje uwierzytelniające.

MQOT_CHANNEL

Kanał.

MQOT_CLNTCONN_CHANNEL

Kanał połączenia klienta.

MQOT_LISTENER

Obiekt nasłuchiwania.

MQOT_NAMELIST,

Lista nazw.

MQOT_PROCESS

Definicja procesu.

Kolejka MQOT_Q

do kolejki błędów.

MQOT_Q_MGR

menedżerze kolejek.

Usługa MQOT_SERVICE

.

Uprawnienie (MQLONG)

Uprawnienie.

W przypadku parametru **Filter** to pole jest ignorowane.

W przypadku parametru **AuthorityBuffer** to pole reprezentuje autoryzacje, które jednostka ma do obiektów zidentyfikowanych przez produkty **ProfileName** i **ObjectType**. Jeśli jednostka ma tylko jedno uprawnienie, to pole jest równe odpowiedniej wartości autoryzacji (stała MQZAO_ *). Jeśli jednostka ma więcej niż jeden organ, to pole jest bitowe OR odpowiednich stałych MQZAO_ *.

EntityDataPtr (PMQZED)

Adres struktury MQZED identyfikujący obiekt.

W przypadku parametru **Filter** pole to wskazuje na strukturę MQZED identyfikującą jednostkę, z której wymagane są dane uprawnień. Jeśli parametr **EntityDataPtr** jest wskaźnikiem wartości NULL, zwracane są dane uprawnień dla wszystkich obiektów.

W przypadku parametru **AuthorityBuffer** pole to wskazuje na strukturę MQZED, która identyfikuje jednostkę, z której pochodzą zwrócone dane uprawnień.

EntityType (MQLONG)

Typ jednostki.

W przypadku parametru **Filter** to pole określa typ jednostki, dla której wymagane są dane uprawnień. Jeśli wartością jest MQZAET_NONE, zwracane są dane o uprawnieniach dla wszystkich typów jednostek.

W przypadku parametru **AuthorityBuffer** to pole określa typ jednostki identyfikowanej przez strukturę MQZED wskazanej przez produkt **EntityDataPtr**.

Wartość ta jest jedną z następujących wartości: dla parametru **Filter** wartość MQZAET_NONE jest również poprawna:

MQZAET_PRINCIPAL

Jednostka główna.

MQZAET_GROUP

Grupa.

Opcje (MQAUTHOPT)

Opcje.

To pole określa opcje, które dają kontrolę nad wyświetlanym profilem.

Należy podać jedną z następujących wartości:

MQAUTHOPT_NAME_ALL_MATCHING

Wyświetla wszystkie profile.

MQAUTHOPT_NAME_EXPLICIT

Wyświetla profile o dokładnie takiej samej nazwie, jak nazwa podana w polu **ProfileName**.

Ponadto należy również określić jeden z następujących elementów:

MQAUTHOPT_ENTITY_SET

Wyświetl wszystkie profile używane do obliczenia skumulowanego uprawnienia, które jednostka ma do obiektu określonego przez **ProfileName**. Pole **ProfileName** nie może zawierać żadnych znaków wieloznacznych.

- Jeśli określony obiekt jest nazwą użytkownika, dla każdego elementu zestawu {entity, groups} zostanie wyświetlony najbardziej odpowiedni profil, który ma zastosowanie do obiektu.
- Jeśli określony obiekt jest grupą, wyświetlany jest najbardziej odpowiedni profil z grupy, która ma zastosowanie do obiektu.
- Jeśli ta wartość zostanie określona, wówczas wartości parametrów **ProfileName**, **ObjectType**, **EntityType** oraz nazwy jednostki określonej w strukturze MQZED produktu **EntityDataPtr** muszą być niepuste.

Jeśli określono parametr **MQAUTHOPT_NAME_ALL_MATCHING**, można również określić następujące informacje:

MQAUTHOPT_ENTITY_EXPLICIT

Wyświetla profile, które mają dokładnie taką samą nazwę obiektu, jak nazwa jednostki określona w strukturze MQZED produktu **EntityDataPtr**.

Deklaracja C

```
typedef struct tagMQZAD MQZAD;  
struct tagMQZAD {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    MQCHAR48   ProfileName;      /* Profile name */  
    MQLONG     ObjectType;        /* Object type */  
    MQLONG     Authority;         /* Authority */  
    PMQZED     EntityDataPtr;    /* Address of MQZED structure identifying an  
                                entity */  
    MQLONG     EntityType;        /* Entity type */  
    MQAUTHOPT  Options;          /* Options */  
};
```

MQZED-deskryptor jednostki

Struktura MQZED jest używana w wielu wywołaniach usługi autoryzacji w celu określenia jednostki, dla której ma zostać sprawdzona autoryzacja.

Pola

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość jest następująca:

MQZED_STRUC_ID

Identyfikator struktury deskryptora jednostki.

W przypadku języka programowania C zdefiniowana jest również stała MQZED_STRUC_ID_ARRAY; ma taką samą wartość jak MQZED_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe dla usługi.

Wersja (MQLONG)

Numer wersji struktury.

Wartość jest następująca:

MQZED_VERSION_1

Struktura deskryptora jednostki Version-1.

Następująca stała określa numer wersji bieżącej wersji:

MQZED_CURRENT_VERSION

Bieżąca wersja struktury deskryptora jednostki.

To jest pole wejściowe dla usługi.

EntityNamePtr (PMQCHAR)

Adres nazwy jednostki.

Jest to wskaźnik do nazwy obiektu, którego autoryzacja ma zostać sprawdzona.

EntityDomainPtr (PMQCHAR)

Adres nazwy domeny jednostki.

Jest to wskaźnik do nazwy domeny zawierającej definicję obiektu, którego autoryzacja ma zostać sprawdzona.

SecurityId (MQBYTE40)

Identyfikator zabezpieczeń.

Jest to identyfikator zabezpieczeń, którego autoryzacja ma zostać sprawdzona.

CorrelationPtr (MQPTR)

Wskaźnik korelacji.

Ułatwia to przekazywanie danych korelacyjnych między funkcją uwierzytelniania użytkownika a innymi odpowiednimi funkcjami OAM.

Deklaracja C

```
typedef struct tagMQZED MQZED;  
struct tagMQZED {  
    MQCHAR4    StrucId;           /* Structure identifier */  
    MQLONG     Version;          /* Structure version number */  
    PMQCHAR    EntityNamePtr;    /* Address of entity name */  
    PMQCHAR    EntityDomainPtr;  /* Address of entity domain name */  
    MQBYTE40   SecurityId;       /* Security identifier */  
    MQPTR      CorrelationPtr;   /* Address of correlation data */  
};
```

MQZFP-Wolne parametry

Ten parametr określa dane związane z zasobem, który ma zostać zwolniony.

Struktura MQZFP jest używana w wywołaniu MQZ_FREE_USER dla parametru *FreeParms*.

Pola

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość jest następująca:

MQZFP_STRUC_ID

Identyfikator struktury wolnych parametrów.

W przypadku języka programowania C zdefiniowana jest również stała MQZFP_STRUC_ID_ARRAY; ma taką samą wartość jak MQZFP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe dla usługi.

Wersja (MQLONG)

Numer wersji struktury.

Wartość jest następująca:

MQZFP_VERSION_1

Struktura wolnych parametrów Version-1.

Następująca stała określa numer wersji bieżącej wersji:

MQZFP_CURRENT_VERSION

Bieżąca wersja struktury wolnych parametrów.

To jest pole wejściowe dla usługi.

Zarezerwowane (MQBYTE8)

Zarezerwowane pole.

Początkowa wartość jest równa null.

CorrelationPtr (MQPTR)

Wskaźnik korelacji.

Adres danych korelacji odnoszących się do zasobu, który ma zostać zwolniony.

Deklaracja C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQBYTE8    Reserved;        /* Reserved field */
    MQPTR      CorrelationPtr;   /* Address of correlation data */
};
```

MQZIC-kontekst tożsamości

Struktura MQZIC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru *IdentityContext*.

Struktura MQZIC zawiera informacje o kontekście tożsamości, które identyfikują użytkownika aplikacji, która po raz pierwszy umiała komunikat w kolejce:

- Menedżer kolejek wypełnia pole UserIdentifier nazwą identyfikującą użytkownika, a sposób działania menedżera kolejek zależy od środowiska, w którym działa aplikacja.
- Menedżer kolejek wypełnia pole AccountingToken znacznikiem lub numerem określonym w aplikacji, w której znajduje się komunikat.
- Aplikacje mogą używać pola ApplIdentityData, aby uzyskać dodatkowe informacje, które mają zostać dołączone do użytkownika (na przykład zaszyfrowane hasło).

Odpowiednio autoryzowane aplikacje mogą ustawiać kontekst tożsamości przy użyciu funkcji MQZ_AUTHENTICATE_USER.

Identyfikator zabezpieczeń systemu Windows (SID) jest przechowywany w polu AccountingToken, gdy komunikat jest tworzony w ramach produktu IBM MQ for Windows. Identyfikator SID może zostać użyty do uzupełnienia pola UserIdentifier oraz do określenia informacji autoryzacyjnych użytkownika.

Pola

StrucId (MQCHAR4)

Identyfikator struktury.

Wartość jest następująca:

MQZIC_STRUC_ID

Identyfikator struktury kontekstu tożsamości.

W przypadku języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY; ta sama wartość ma wartość MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

To jest pole wejściowe dla usługi.

Wersja (MQLONG)

Numer wersji struktury.

Wartość jest następująca:

MQZIC_VERSION_1

Struktura kontekstu tożsamości Version-1.

Następująca stała określa numer wersji bieżącej wersji:

MQZIC_CURRENT_VERSION

Bieżąca wersja struktury kontekstu tożsamości.

To jest pole wejściowe dla usługi.

UserIdentifier (MQCHAR12)

Identyfikator użytkownika.

Jest to część **kontekstu tożsamości** komunikatu.

UserIdentifier określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje formatu tego pliku. Więcej

informacji na temat pola *UserIdentifier* zawiera sekcja [“UserIdentifier \(MQCHAR12\)”](#) na stronie 465.

AccountingToken (MQBYTE32)

Token rozliczania.

Jest to część **kontekstu tożsamości** komunikatu.

Produkt *AccountingToken* umożliwia aplikacji wykonanie pracy wykonanej w wyniku komunikatu, który ma być odpowiednio obciążony. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza jego zawartości. Więcej informacji na temat pola *AccountingToken* zawiera sekcja [“AccountingToken \(MQBYTE32\)”](#) na stronie 420.

Dane ApplIdentity(MQCHAR32)

Dane aplikacji odnoszące się do tożsamości.

Jest to część **kontekstu tożsamości** komunikatu.

ApplIdentityData to informacje zdefiniowane przez pakiet aplikacji, które mogą być używane do udostępniania dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiona przez aplikacje działające z odpowiednim uprawnieniem użytkownika w celu wskazania, czy dane tożsamości są zaufane. Więcej informacji na temat pola *ApplIdentityData* zawiera sekcja [“Dane ApplIdentity\(MQCHAR32\)”](#) na stronie 422.

Deklaracja C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
    MQCHAR4    StrucId;           /* Structure identifier */
    MQLONG     Version;          /* Structure version number */
    MQCHAR12   UserIdentifier;    /* User identifier */
    MQBYTE32   AccountingToken;  /* Accounting token */
    MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

Materiał odniesienia dla IBM MQ bridge for HTTP

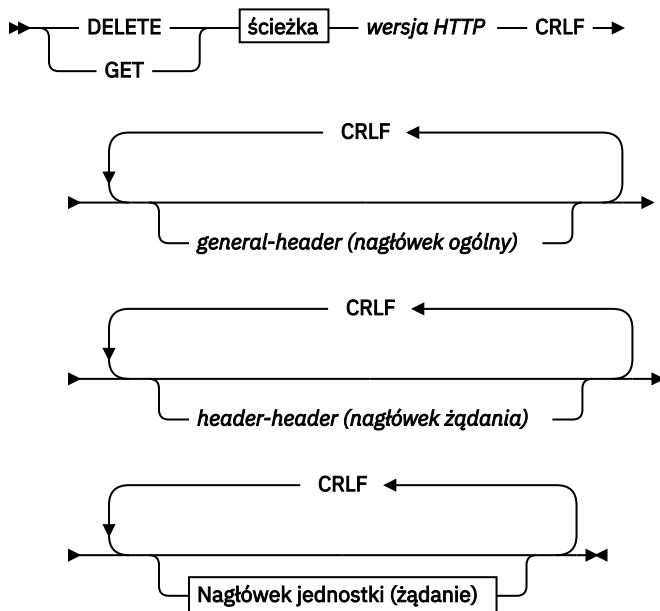
Tematy dotyczące produktu IBM MQ bridge for HTTP, uporządkowane alfabetycznie

HTTP DELETE: komenda IBM MQ bridge for HTTP

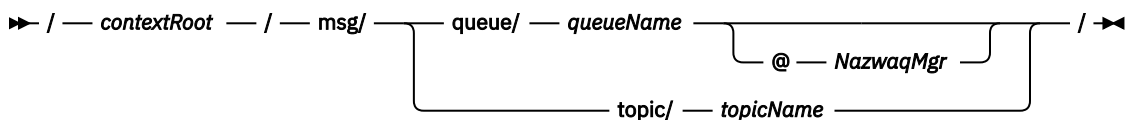
Operacja HTTP **DELETE** pobiera komunikat z kolejki produktu IBM MQ lub pobiera publikację z tematu. Komunikat jest usuwany z kolejki. Jeśli publikacja zostanie zachowana, nie zostanie usunięta. Komunikat odpowiedzi jest przesyłany z powrotem do klienta, w tym do informacji o komunikacie.

Składnia

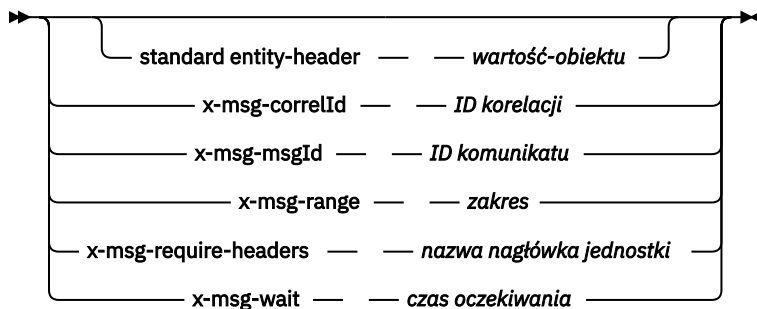
Żądanie



Ścieżka



entity-header (Żądanie)

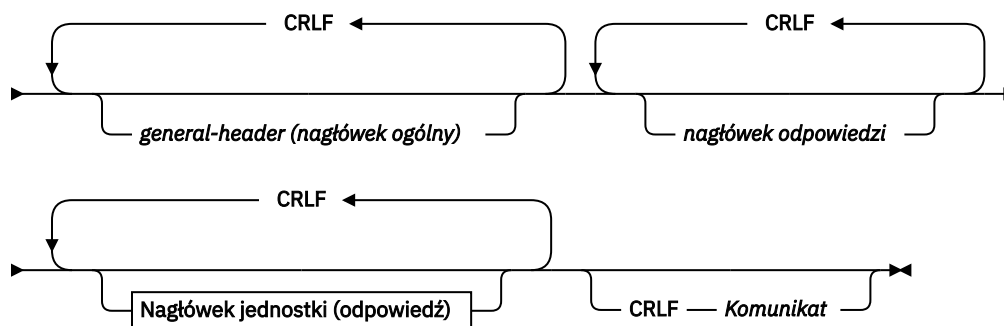


Uwaga:

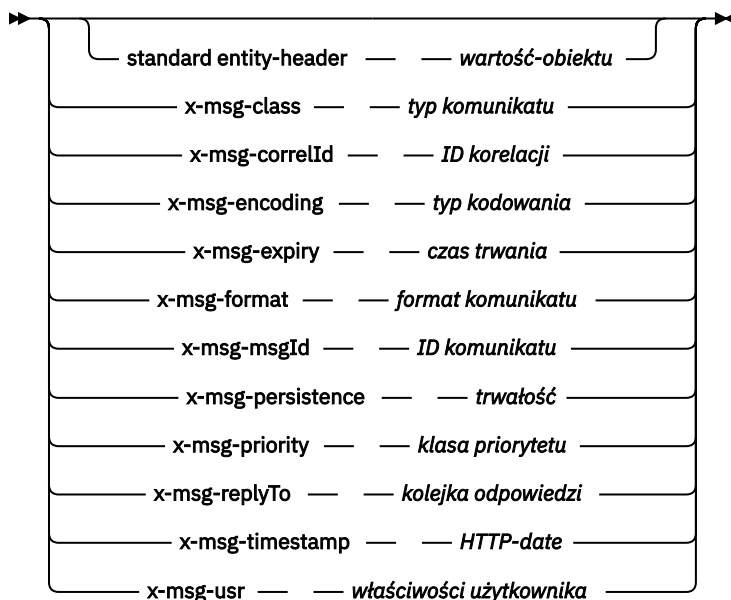
1. Jeśli używany jest znak zapytania (?), należy go zastąpić znakiem zapytania %3f. Na przykład wartość orange?topic powinna być określona jako orange%3ftopic.
2. @ qMgrNazwa jest poprawny tylko na serwerze HTTP **POST**

Działanie

➡ *Wersja HTTP* — — *Status HTTP-Kod* — — *Fraza przyczyny HTTP* — CRLF ➡



entity-header (Odpowiedź)



Parametry żądania

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz HTTP/1.1 - 4.5 Ogólne Pola nagłówka.

nagłówek-żądania

Patrz HTTP/1.1 - 5.3 -Pola nagłówka żądania. Pole Host jest obowiązkowe w żądaniu HTTP/1.1 . Często jest on automatycznie wstawiany za pomocą narzędzia, które jest używane do tworzenia żądania klienta.

entity-header (żądanie)

Patrz HTTP/1.1 - 7.1 -Pola nagłówka jednostki. Jeden z nagłówków jednostek wymienionych w diagramie składniowym żądania.

Parametry odpowiedzi

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz [HTTP/1.1 - 4.5 Ogólne Pola nagłówka](#).

nagłówek-odpowiedzi

Patrz [HTTP/1.1 - 6.2 -Pola Nagłówka Odpowiedzi](#).

entity-header (odpowiedź)

Patrz [HTTP/1.1 - 7.1 -Pola nagłówka jednostki](#). Jedna z nagłówków obiektu lub odpowiedzi wymienionych w diagramie składni odpowiedzi. Wartość `Długość - treści` jest zawsze obecna w odpowiedzi. Wartość ta jest ustawiona na zero, jeśli nie ma treści komunikatu.

komunikat

Treść komunikatu.

Opis

Jeśli żądanie HTTP **DELETE** zakończy się pomyślnie, komunikat odpowiedzi będzie zawierał dane pobrane z kolejki produktu IBM MQ. Liczba bajtów w treści komunikatu jest zwracana w nagłówku HTTP `Content - Length`. Kod statusu dla odpowiedzi HTTP jest ustawiony na wartość 200 OK. Jeśli wartość `x -msg - range` jest określona jako 0lub 0-0, to kod statusu odpowiedzi HTTP to 204 No Content.

Jeśli żądanie HTTP **DELETE** nie powiedzie się, odpowiedź będzie zawierać komunikat o błędzie IBM MQ bridge for HTTP oraz kod statusu HTTP.

Przykład metody HTTP DELETE

Metoda HTTP **DELETE** umożliwia pobranie komunikatu z kolejki i jego usunięcie lub pobranie i usunięcie publikacji. Przykład komendy **HTTPDELETE** języka Java to przykład, w którym wykonywany jest odczyt komunikatu z kolejki przez żądanie HTTP **DELETE**. Zamiast używać języka Java, można utworzyć żądanie HTTP **DELETE** przy użyciu formularza przeglądarki lub przybornika AJAX.

Poniższy rysunek przedstawia żądanie HTTP, aby usunąć następny komunikat w kolejce o nazwie `myQueue`. W odpowiedzi treść komunikatu jest zwracana do klienta. W kontekście produktu IBM MQ metoda HTTP **DELETE** jest niszczącą operacją pobierania.

Żądanie zawiera nagłówek żądania HTTP `x -msg -wait`, który stanowi instrukcję mostu IBM MQ dla protokołu HTTP dotyczącą długości oczekiwania na przybycie komunikatu do kolejki. Żądanie zawiera również nagłówek żądania `x -msg -require -headers`, który wskazuje, że klient ma otrzymać w odpowiedzi identyfikator korelacji komunikatu.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Rysunek 37. Przykład żądania HTTP **DELETE***

Poniższy rysunek przedstawia odpowiedź zwróconą do klienta. Identyfikator korelacji jest zwracany do klienta zgodnie z żądaniem w nagłówku `x -msg -require -headers` żądania.

```

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.

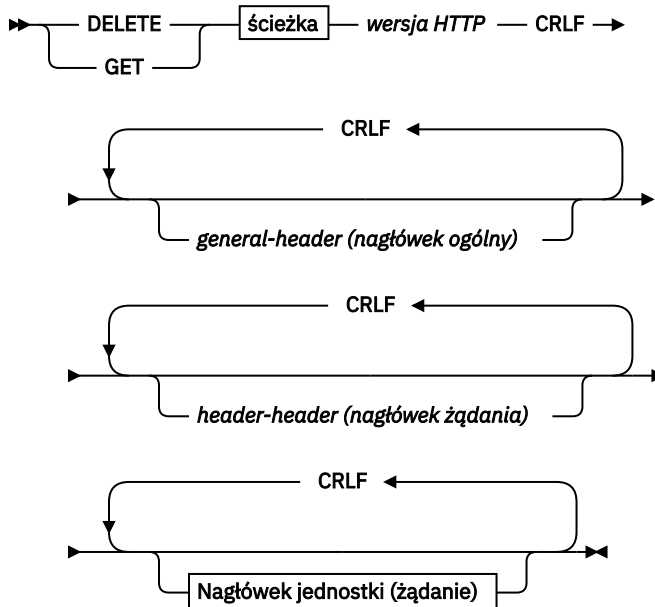
Rysunek 38. Przykład odpowiedzi HTTP **DELETE**

HTTP GET: komenda IBM MQ bridge for HTTP

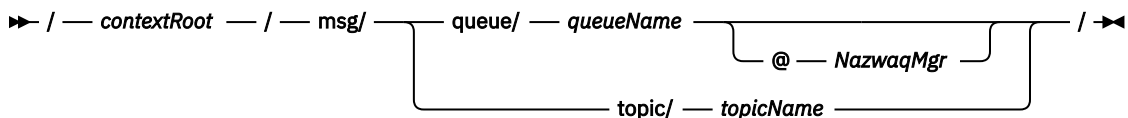
Operacja HTTP **GET** pobiera komunikat z kolejki produktu IBM MQ . Komunikat pozostaje w kolejce.
Operacja HTTP **GET** jest równoznaczna z przeglądaniem kolejki produktu IBM MQ .

Składnia

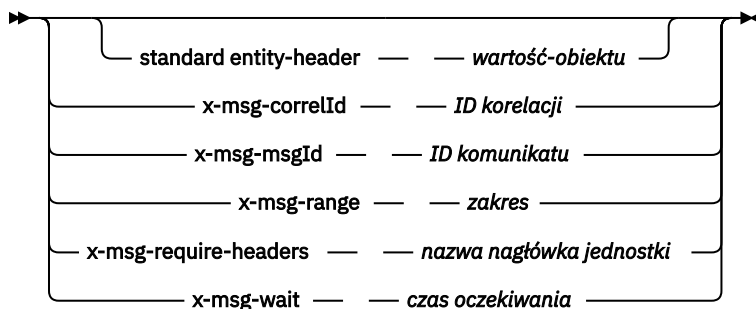
Żądanie



Ścieżka



entity-header (Żądanie)

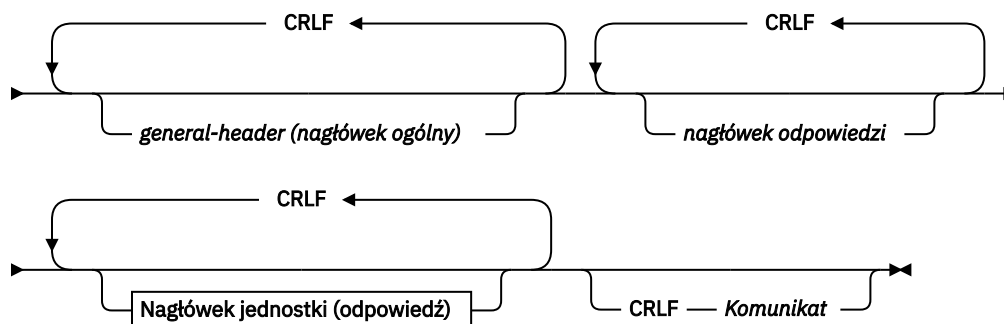


Uwaga:

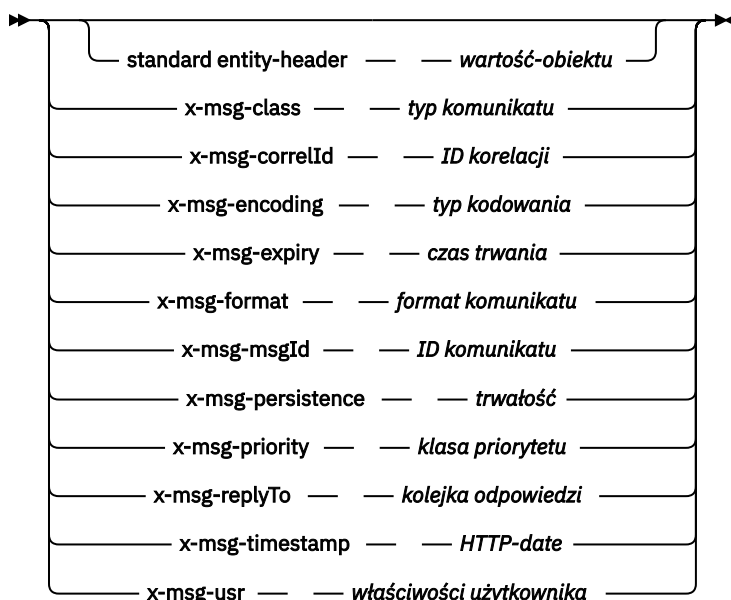
1. Jeśli używany jest znak zapytania (?), należy go zastąpić znakiem zapytania %3f. Na przykład wartość `orange?topic` powinna być określona jako `orange%3ftopic`.
2. `@qMgrNazwa` jest poprawny tylko na serwerze HTTP **POST**

Działanie

➔ *Wersja HTTP* — — *Status HTTP-Kod* — — *Fraza przyczyny HTTP* — CRLF ➔



entity-header (Odpowiedź)



Parametry żądania

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz [HTTP/1.1 - 4.5 Ogólne Pola nagłówka](#).

nagłówek-żądania

Patrz [HTTP/1.1 - 5.3 -Pola nagłówka żądania](#). Pole Host jest obowiązkowe w żądaniu HTTP/1.1 . Często jest on automatycznie wstawiany za pomocą narzędzia, które jest używane do tworzenia żądania klienta.

entity-header (żądanie)

Patrz [HTTP/1.1 - 7.1 -Pola nagłówka jednostki](#). Jeden z nagłówków jednostek wymienionych w diagramie składniowym żądania.

Parametry odpowiedzi

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz [HTTP/1.1 - 4.5 Ogólne Pola nagłówka](#).

nagłówek-odpowiedzi

Patrz [HTTP/1.1 - 6.2 -Pola Nagłówka Odpowiedzi](#).

entity-header (odpowiedź)

Patrz [HTTP/1.1 - 7.1 -Pola nagłówka jednostki](#). Jedna z nagłówków obiektu lub odpowiedzi wymienionych w diagramie składni odpowiedzi. Wartość *Długość - treści* jest zawsze obecna w odpowiedzi. Wartość ta jest ustawiona na zero, jeśli nie ma treści komunikatu.

komunikat

Treść komunikatu.

Opis

Jeśli żądanie HTTP **GET** zakończy się pomyślnie, komunikat odpowiedzi będzie zawierał dane pobrane z kolejki produktu IBM MQ . Liczba bajtów w treści komunikatu jest zwracana w nagłówku HTTP Content - Length . Kod statusu dla odpowiedzi HTTP jest ustawiony na wartość 200 OK. Jeśli wartość x -msg -range jest określona jako 0lub 0-0, to kod statusu odpowiedzi HTTP to 204 No Content.

Jeśli żądanie HTTP **GET** nie powiedzie się, odpowiedź będzie zawierać komunikat o błędzie IBM MQ bridge for HTTP oraz kod statusu HTTP.

Przykład metody HTTP GET

Metoda HTTP **GET** pobiera komunikat z kolejki. Komunikat pozostaje w kolejce. W kontekście produktu IBM MQ metoda HTTP **GET** jest żądaniem przeglądania. Żądanie metody HTTP **GET** można utworzyć przy użyciu klienta Java, formularza przeglądarki lub przybornika AJAX.

Poniższy rysunek przedstawia żądanie HTTP, aby przejrzeć następny komunikat w kolejce o nazwie myQueue.

Żądanie zawiera nagłówek żądania HTTP x -msg -wait, który stanowi instrukcję mostu IBM MQ dla protokołu HTTP dotyczącą długości oczekiwania na przybycie komunikatu do kolejki. Żądanie zawiera również nagłówek żądania x -msg -require -headers, który wskazuje, że klient ma otrzymać w odpowiedzi identyfikator korelacji komunikatu.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

Rysunek 39. Przykład żądania HTTP GET

Poniższy rysunek przedstawia odpowiedź zwróconą do klienta. Identyfikator korelacji jest zwracany do klienta zgodnie z żądaniem w nagłówku x -msg -require -headers żądania.

```

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.

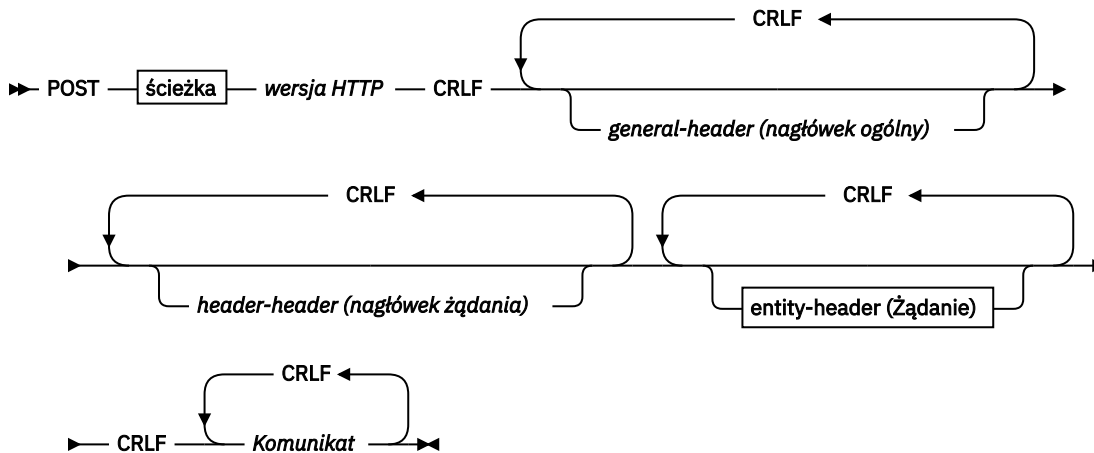
Rysunek 40. Przykład odpowiedzi HTTP **GET**

HTTP POST: komenda IBM MQ bridge for HTTP

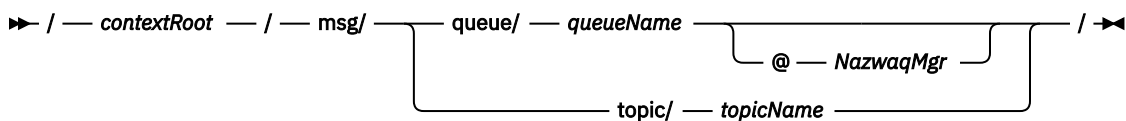
Operacja HTTP **POST** umieszcza komunikat w kolejce produktu IBM MQ lub publikuje komunikat w temacie.

Składnia

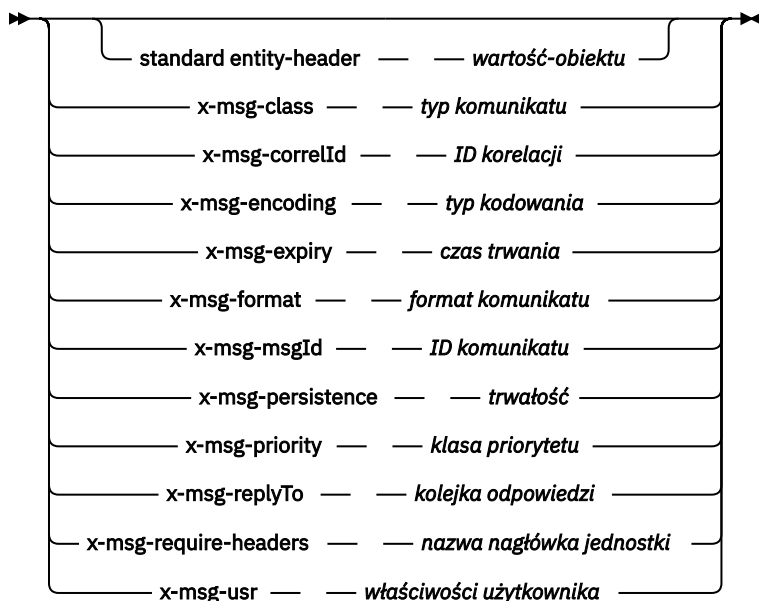
Żądanie



Ścieżka



entity-header (Żądanie)

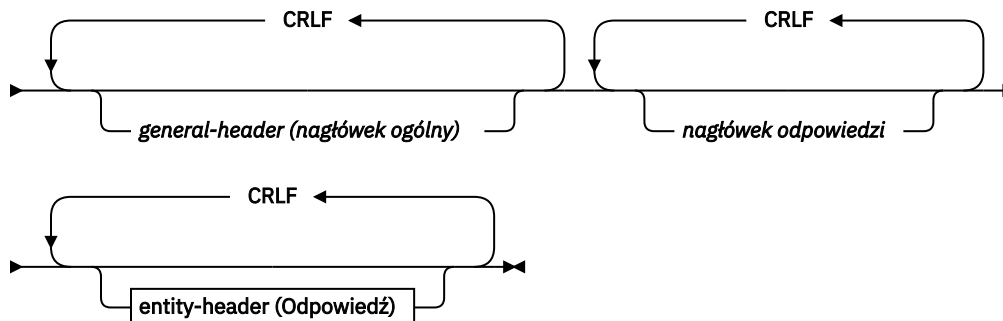


Uwaga:

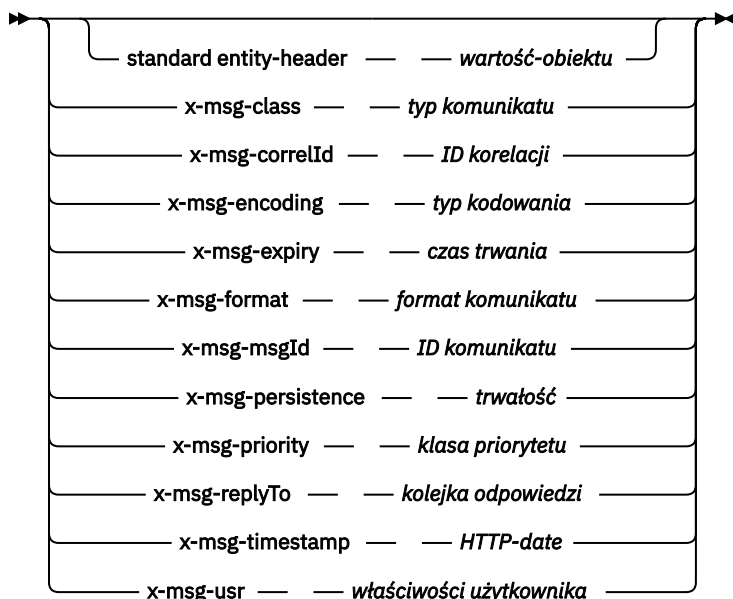
1. Jeśli używany jest znak zapytania (?), należy go zastąpić znakiem zapytania %3f. Na przykład wartość orange?topic powinna być określona jako orange%3ftopic.
2. @ qMgrNazwa jest poprawny tylko na serwerze HTTP **POST**

Działanie

➡ Wersja HTTP — — Status HTTP-Kod — — Fraza przyczyny HTTP — CRLF ➡



entity-header (Odpowiedź)



Parametry żądania

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz [HTTP/1.1 - 4.5 Ogólne Pola nagłówka](#).

nagłówek-żądania

Patrz [HTTP/1.1 - 5.3 -Pola nagłówka żądania](#). Pole Host jest obowiązkowe w żądaniu HTTP/1.1 . Często jest on automatycznie wstawiany za pomocą narzędzia, które jest używane do tworzenia żądania klienta.

entity-header (żądanie)

Patrz [HTTP/1.1 - 7.1 -Pola nagłówka jednostki](#). Jeden z nagłówków jednostek wymienionych w diagramie składni żądania. W żądaniu powinny zostać wstawione elementy Content-Length i Content-Type , które często są wstawiane automatycznie przez narzędzie używane do tworzenia żądania klienta. Typ Content-Type musi być zgodny z typem zdefiniowanym w niestandardowym nagłówku encji x-msg-class , jeśli jest on określony.

komunikat

Komunikat do umieszczenia w kolejce lub opublikowanie w celu opublikowania tematu.

Parametry odpowiedzi

Ścieżka

Patrz sekcja [“Format identyfikatora URI”](#) na stronie 1855.

Wersja HTTP

Wersja protokołu HTTP, na przykład HTTP/1.1

nagłówek ogólny

Patrz [HTTP/1.1 - 4.5 Ogólne Pola nagłówka](#).

nagłówek-odpowiedzi

Patrz [HTTP/1.1 - 6.2 -Pola Nagłówka Odpowiedzi](#).

entity-header (odpowiedź)

Patrz HTTP/1.1 - 7.1 - Pola nagłówka jednostki. Jedna z nagłówków obiektu lub odpowiedzi wymienionych w diagramie składni odpowiedzi. Wartość Długość - treści jest zawsze obecna w odpowiedzi. Wartość ta jest ustawiona na zero, jeśli nie ma treści komunikatu.

Opis

Jeśli żaden nagłówek `x-msg-usr` nie zostanie uwzględniony, a klasa komunikatu to `BAJTY` lub `TEKST`, komunikat umieszczony w kolejce nie ma `MQRFH2`.

Użyj jednostki HTTP i nagłówków żądania w żądaniu HTTP **POST**, aby ustawić właściwości komunikatu umieszczonego w kolejce. Można również użyć opcji `x-msg-require-headers`, aby zażądać, które nagłówki zostaną zwrócone w komunikacie odpowiedzi.

Jeśli żądanie HTTP **POST** zakończy się pomyślnie, to obiekt komunikatu odpowiedzi jest pusty, a jego wartość `Content-Length` wynosi zero. Kod statusu HTTP: 200 OK.

Jeśli żądanie HTTP **POST** nie powiedzie się, odpowiedź będzie zawierać komunikat o błędzie IBM MQ bridge for HTTP oraz kod statusu HTTP. Komunikat IBM MQ nie jest umieszczany w kolejce ani w temacie.

Przykład metody HTTP POST

Metoda HTTP **POST** umożliwia umieszczenie komunikatu w kolejce lub jego opublikowanie w temacie. Przykład komendy **HTTPPOST** Java to przykład żądania HTTP **POST** dotyczącego umieszczenia komunikatu w kolejce. Zamiast używać języka Java, można utworzyć żądanie HTTP **POST** przy użyciu formularza przeglądarki lub przybownika AJAX.

Poniższy rysunek przedstawia żądanie HTTP dotyczące umieszczenia komunikatu w kolejce o nazwie `myQueue`. To żądanie zawiera nagłówek HTTP `x-msg-correlID` umożliwiający ustawienie identyfikatora korelacji komunikatu produktu IBM MQ.

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50
```

Here is my message body that is posted on the queue.

*Rysunek 41. Przykład żądania HTTP **POST** w kolejce*

Poniższy rysunek przedstawia odpowiedź odesłaną do klienta. Brak treści odpowiedzi.

```
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
```

*Rysunek 42. Przykład odpowiedzi HTTP **POST***

Nagłówki HTTP

Produkt IBM MQ bridge for HTTP obsługuje niestandardowe nagłówki HTTP żądań, niestandardowe nagłówki HTTP jednostki oraz podzbiór standardowych nagłówków HTTP.

Praktyka HTTP polega na przedrostku wszystkich nagłówków niestandardowych z produktem `x-`, dlatego dla nagłówków HTTP IBM MQ Bridge dla nagłówków HTTP jest poprzedzona przedrostkiem `x-msg-`. Na przykład, aby ustawić nagłówek priorytetu, użyj komendy `x-msg-priority`.

Uwaga:

- W większości wartości nagłówek rozróżniana jest wielkość liter. Na przykład, gdy używany jest nagłówek `msgId`, `NONE` jest słowem kluczowym, natomiast `none` to `msgID`.
- Błędnie wpisane nagłówki są ignorowane.

Nagłówki HTTP jednostki niestandardowej

Nagłówki HTTP jednostki niestandardowej zawierają informacje na temat komunikatów produktu IBM MQ. Za pomocą nagłówków obiektów można ustawiać wartości w deskrytorze komunikatu (MQMD) lub wartości zapytań w MQMD. Dodatkowy nagłówek jednostki, `x-msg-usr`, ustawia i zwraca wszystkie informacje o właściwościach użytkownika, które mają zostać powiązane z żądaniem.

Nagłówki jednostek można używać w różnych kontekstach żądań HTTP:

DELETE

W przypadku żądania HTTP **DELETE** można używać tylko nagłówków `x-msg-correlId`, `x-msg-msgId` lub obu. Efektem tych nagłówków jest wybranie określonego komunikatu przez element `MsgId` i `CorrelId` w MQGET oraz usunięcie komunikatu z jego kolejki.

GET

W przypadku żądania HTTP **GET** można używać tylko nagłówków `x-msg-correlId`, `x-msg-msgId` lub obu. Efektem tych nagłówków jest wybranie określonego komunikatu przez element `MsgId` i element `CorrelId` w polu MQGET for browse.

POST

W żądaniu HTTP **POST** można użyć dowolnego nagłówka jednostki, z wyjątkiem `x-msg-timestamp`.

x-msg-require-headers

W przypadku dowolnego żądania HTTP **GET**, **POST** lub **DELETE** można dodać wiele nagłówków jednostek wewnątrz nagłówka żądania `x-msg-require-headers`, rozdzielając je przecinkami. Efektem jest zwrócenie określonych nagłówków jednostek w komunikacie odpowiedzi HTTP, które zawiera wartość powiązanej właściwości komunikatu.

Opis wszystkich list nagłówków, w których nagłówek jest przetwarzany przez produkt IBM MQ bridge for HTTP. Na przykład w nagłówku **POST**, `x-msg-require-headers`, nagłówek jest przetwarzany przez IBM MQ bridge for HTTP w żądaniu HTTP **POST** lub w nagłówku żądania `x-msg-require-headers` w żądaniu HTTP **POST**, **GET** lub **DELETE**. Jeśli nagłówek jest uwzględniany w kontekście, w którym nie jest dozwolony, nagłówek jest ignorowany. Nie zgłaszono żadnego błędu.

Można umieścić wszystkie standardowe nagłówki HTTP w żądaniach, które mają być przetwarzane przez serwer WWW, lub inne procedury obsługi żądań. Podobnie odpowiedź może zawierać inne standardowe nagłówki HTTP wstawione przez serwer WWW lub inne procedury obsługi odpowiedzi.

Nagłówki HTTP żądania niestandardowego

Trzy niestandardowe nagłówki HTTP żądania, `x-msg-range`, `x-msg-require-headers` i `x-msg-wait`, przekazują dodatkowe informacje o żądaniu HTTP do serwera. Działają one jako modyfikatory żądań. W przypadku parametru `x-msg-range` można ograniczyć ilość danych komunikatu zwracanych w odpowiedzi. Za pomocą opcji `x-msg-require-headers` można zażądać odpowiedzi, aby zawierała informacje na temat wyniku żądania. Za pomocą komendy `x-msg-wait` można zmodyfikować czas oczekiwania klienta na odpowiedź HTTP.

Standardowe nagłówki HTTP

Standardowy nagłówek żądania HTTP `Host` musi być określony w żądaniu HTTP/1.1.

W żądaniu można określić standardowe nagłówki encji HTTP `Content-Length` i `Content-Type`.

W odpowiedzi na żądanie mogą zostać zwrócone standardowe nagłówki encji HTTP: `Content-Length`, `Content-Location`, `Content-Range`, `Content-Type` i `Server`. Określ jeden lub więcej standardowych nagłówków HTTP w nagłówku `x-msg-request-header` w komunikacie żądania.

Alfabetyczna lista nagłówków HTTP

class: nagłówek encji HTTP x-msg-class

Ustaw lub zwróć typ komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	klasa x-msg
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST, x-msg-require-headers
Dozwolone wartości	BYTES MAP STREAM TEXT
Wartość domyślna	BYTES

Opis

- W żądaniu HTTP **POST** ustawia typ utworzonego komunikatu.
- Określenie nagłówka klasy na serwerze **GET** lub **DELETE** zwraca 400 Bad Request z treścią jednostki MQHTTP40007.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-class w komunikacie odpowiedzi HTTP na typ komunikatu.
- Jeśli dla tego nagłówka zostanie podana niepoprawna wartość, zwracany jest komunikat MQHTTP40005.
- Jeśli nagłówek x-msg-class nie zostanie podany, a typem treści komunikatu jest application/x-www-form-urlencoded, przyjmuje się, że dane są obiektem mapy JMS.

Content-Length: encja HTTP-nagłówek

Ustawia lub zwraca długość (w bajtach) treści komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	Content-Length (długość treści)
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	x-msg-require-headers
Wartość dozwolona i zwrócona	Integer value Długość (w bajtach) treści komunikatu.

Opis

- Opcja Content-Length jest opcjonalna w żądaniu HTTP. W przypadku wartości **GET** lub **DELETE** długość musi wynosić zero. W przypadku produktu **POST**, jeśli podano opcję Content-Length, a nie jest ona zgodna z długością wiersza komunikatu, komunikat jest obcinany lub dopełniany wartościami null na określoną długość.
- Wartość Content-Length jest zawsze zwracana w odpowiedzi HTTP, nawet wtedy, gdy nie ma treści, w którym to przypadku wartość jest równa zero.

Content-Location: obiekt HTTP-nagłówek

Zwraca kolejkę lub temat, do którego odwołuje się żądanie, w standardowym nagłówku Content-Location w komunikacie odpowiedzi HTTP.

Typ	Opis
Nazwa nagłówka HTTP	Zawartość-Położenie
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	x-msg-require-headers
Wartość zwracana	Identyfikator URI w formacie, <i>/msg/queue/queueName</i> lub wersji <i>/msg/topic/topicName</i>

Opis

- W przypadku żądania w x-msg-require-headers nagłówek encji Content-Location zwraca kolejkę lub temat, do którego odwołuje się żądanie HTTP.

Content-Range: obiekt HTTP-nagłówek

Zwraca zakres bajtów wybranych z komunikatu IBM MQ w nagłówku Content-Range w odpowiedzi HTTP.

Typ	Opis
Nazwa nagłówka HTTP	Zakres-zawartości
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	x-msg-require-headers
Wartość zwracana	String Zwraca dolny limit, <i>m</i> i górny limit, <i>n</i> zwracanego podłańcucha i <i>length</i> całego komunikatu. Na przykład składnia <i>m - n/length</i>

Opis

-
- Wartość Zakres-zakresu-treści jest zwracana tylko w odpowiedzi HTTP, jeśli w żądaniu **GET** lub **DELETE**, który zawiera nagłówek żądania x-msg-range, określono wartość Zakres-zakresu.
- Jeśli wartość x-msg-range jest określona w żądaniu **GET** lub **DELETE**, to zakres bajtów określony w nagłówku Content-Range jest zwracany w odpowiedzi. Na przykład, jeśli w żądaniu komunikatu zawierającego 100 bajtów używany jest produkt x-msg-range: 0-60, nagłówek content-range zawiera łańcuch 0-60/100.
- Żądanie x-msg-range zwraca również zakres treści w nagłówku x-msg-range w odpowiedzi HTTP.

Content-Type: obiekt HTTP-nagłówek

Ustaw lub zwracaj klasę komunikatu JMS w komunikacie IBM MQ zgodnie z typem treści HTTP.

Typ	Opis
Nazwa nagłówka HTTP	Content-Type
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolona lub zwracana wartość	media-type W przypadku typów nośników, które są obsługiwane, patrz Tabela 212 na stronie 1833 .

Tabela 212. Odzworowanie między elementami x-msg-class i HTTP Content-Type .

klasa x-msg	HTTP Content-Type
BYTES	application/octet-stream application/xml
TEXT	text/*
MAP	application/x-www-form-urlencoded application/xml (opcjonalny).
STREAM	application/xml (opcjonalny).

Opis

- W żądaniu HTTP **POST** określ wartość Content-Type lub x-msg-class. If you specify both, they must be consistent or an HTTP Bad Request exception, Status code 400 is returned. W przypadku pominięcia obu wartości: Content-Type i x-msg-class, przyjmowana jest wartość Content-Type z text/*.
- Typ Content-Type jest zawsze ustawiany w odpowiedzi na HTTP **GET** lub **DELETE**, które mają treść komunikatu. Typ Content-Type jest ustawiany zgodnie z regułami w produkcie [Tabela 213 na stronie 1833](#).

Tabela 213. Odzworowywanie typów komunikatów na x-msg-class i Content-Type

Format wiadomości	JMS Typ komunikatu	klasa x-msg	Content-Type
Wszystko oprócz MQFMT_STRING	Brak	BYTES	application/octet-stream
MQFMT_STRING	Brak	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

Ustaw lub zwróć identyfikator korelacji.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-correlId
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	DELETE, GET, POST , x-msg-require-headers
Dozwolone wartości	String value Na przykład: <pre>x-msg-correlId: mycorrelationid</pre> Dozwolone są łańcuchy ujęte w znaki cudzysłowu, na przykład: <pre>x-msg-correlId: "my id"</pre> Hex value Wartość szesnastkowa poprzedzona przedrostkiem 0x : ; na przykład: <pre>x-msg-correlId: 0x:43c1d23a</pre> Wartość szesnastkowa po 0x: jest ograniczona do 48 znaków reprezentujących 24 bajty. Dodatkowe dane są ignorowane.
Wartość domyślna	Nie dotyczy

Opis

- W przypadku żądania HTTP **POST** ustawia identyfikator korelacji utworzonego komunikatu.
- Na żądanie HTTP **GET** lub **DELETE** wybiera komunikat z kolejki lub tematu. Jeśli nie istnieje żaden komunikat o podanym identyfikatorze korelacji, zwracana jest odpowiedź HTTP 504 Gateway Timeout . x-msg-correlId może być używany z x-msg-msgID , aby wybrać komunikat z kolejki lub tematu, który jest zgodny z obydwojema selektorami.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-coreId w komunikacie odpowiedzi HTTP na identyfikator korelacji komunikatu.
- Po przedrostku 0x : dozwolone jest poziome białe znaki.

Uwaga:

- Określenie wartości x-msg-correlId bez wartości w żądaniu HTTP **GET** lub **DELETE** , na przykład "x-msg-correlId: ", zwraca następny komunikat w kolejce lub temat bez względu na jego identyfikator korelacji.
- Jeśli zostanie określony selektor o długości 24 znaków lub mniejszej, lub 0x: , po którym następują 48 znaków lub mniej, produkt IBM MQ bridge for HTTP użyje zoptymalizowanego selektora w celu zwiększenia wydajności.
- Selektor komunikatów produktu JMS zawierający atrybut JMSCorrelationID jest używany podczas wybierania komunikatów z kolejki. Ten selektor jest zachowywany zgodnie z opisem w sekcji [Zachowanie wyboru](#).

encoding: HTTP x-msg-encoding entity-header

Ustaw lub zwróć kodowanie komunikatów.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-encoding
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	Rozdzielana przecinkami lista następujących wartości: DECIMAL_NORMAL DECIMAL_REVERSED FLOAT_IEEE_NORMAL FLOAT_IEEE_REVERSED FLOAT_S390 INTEGER_NORMAL INTEGER_REVERSED Na przykład składnia <pre>x-msg-encoding: INTEGER_NORMAL,DECIMAL_NORMAL,FLOAT_IEEE_NORMAL</pre> Uwaga: W wartości rozróżniana jest wielkość liter
Wartość domyślna	DECIMAL_NORMAL, FLOAT_IEEE_NORMAL, INTEGER_NORMAL

Opis

- W żądaniu HTTP **POST** określa kodowanie utworzonego komunikatu.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-encoding jest ignorowany.
- Wartość określona w polu x-msg-require-headersustawia właściwość x-msg-encoding w komunikacie odpowiedzi HTTP na właściwość encoding komunikatu.

utrata ważności: encja HTTP x-msg-wygaśnięcie komunikatu -nagłówek

Ustaw lub zwróć czas trwania utraty ważności komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-wygaśnięcie
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	UNLIMITED Na przykład: <pre>x-msg-expiry: UNLIMITED</pre> Integer value Milisekundy przed utratą ważności. Na przykład:

Typ	Opis
	x-msg-expiry: 10000
Wartość domyślna	UNLIMITED

Opis

- W przypadku ustawienia żądania HTTP **POST** komunikat żądania traci ważność w podanym czasie.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-wygaśnięcie jest ignorowany.
- Określone w x-msg-require-headers, ustawia x-msg-wygaśnięcie w komunikacie odpowiedzi HTTP do czasu utraty ważności komunikatu.
- UNLIMITED określa, że komunikat nigdy nie traci ważności.
- Wygaśnięcie komunikatu rozpoczyna się od momentu nadejścia komunikatu do kolejki, ponieważ opóźnienie sieci wynikowej jest ignorowane.
- Wartość maksymalna jest ograniczona przez IBM MQ do 214748364700 milisekund. Jeśli wartość jest większa niż wartość określona, przyjmuje się, że maksymalny czas utraty ważności jest dopuszczalny.

format: obiekt HTTP x-msg-format -nagłówek-nagłówek

Ustaw lub zwróć format komunikatu IBM MQ .

Typ	Opis
Nazwa nagłówka HTTP	x-msg-format
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	NONE Na przykład składnia <pre>x-msg-format: NONE</pre> String value Dowolna wartość zdefiniowana przez użytkownika o długości do ośmiu znaków. Na przykład składnia <pre>x-msg-format: myformat</pre>
Wartość domyślna	None

Opis

- W przypadku ustawienia żądania HTTP **POST** ustaw format komunikatu żądania.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-format jest ignorowany.
- Określone w x-msg-require-headers, ustawia x-msg-format w komunikacie odpowiedzi HTTP na format komunikatu.
- NONE rozróżnia wielkość liter i wskazuje, że format komunikatu jest pusty.
- Używana jest wartość x-msg-format , nawet jeśli jest ona sprzeczna z typem nośnika żądania HTTP. Patrz [Tabela 214 na stronie 1837](#).

Tabela 214. Odzworowanie typu treści i typu *x-msg-class* na format komunikatu

klasa <i>x-msg</i>	Content-Type	Format komunikatu dla kolejki/tematu
BYTES	- application/octet-stream - application/xml	Komunikat IBM MQ : MQFMT ustaw na MQC.MQFMT_NONE
TEXT	text/*	Komunikat IBM MQ : MQFMT ustaw na MQC.MQFMT_STRING
MAP	- application/x-www-form-urlencoded - application/xml (opcjonalny).	JMSMap
STREAM	application/xml (opcjonalny).	JMSStream

msgId: HTTP *x-msg-msgId* entity-header

Ustaw lub zwróć identyfikator komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	<i>x-msg-msgId</i>
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacji żądania HTTP	DELETE, GET, POST , <i>x-msg-require-headers</i>
Dozwolone wartości	String value Na przykład składnia <pre>x-msg-msgId: mymsgid</pre> Łańcuchy ujęte w znaki cudzysłowu, na przykład <i>x-msg-msgId</i>: "my id" Hex value Wartość szesnastkowa poprzedzona przedrostkiem 0x : ; na przykład, <pre>x-msg-msgId: 0x:43c1d23a</pre>
Wartość domyślna	Nie dotyczy

Opis

- W przypadku żądania HTTP **POST** ustawia identyfikator komunikatu utworzonego przez komunikat.
- Na żądanie HTTP **GET** lub **DELETE** wybiera komunikat z kolejki lub tematu. Jeśli z podanym identyfikatorem komunikatu nie istnieje żaden komunikat, zwracana jest odpowiedź HTTP 504 Gateway Timeout. *x-msg-msgId* można użyć z *x-msg-correlID*, aby wybrać komunikat z kolejki lub tematu, który jest zgodny z obydwoma selektorami.
- Wartość podana w parametrze *x-msg-require-headers* zwraca wartość *x-msg-msgId* w odpowiedzi HTTP na identyfikator komunikatu.
- Po przedrostku 0x: dozwolone jest poziome białe znaki.

Uwaga: Określenie parametru x-msg-msgId bez wartości na żądanie HTTP **GET** lub **DELETE** , na przykład "x-msg-msgId: ", zwraca następny komunikat w kolejce lub temat niezależnie od identyfikatora komunikatu.

Selektor komunikatów produktu JMS zawierający element JMSMessageID jest używany podczas wybierania komunikatów z kolejki. Ten selektor jest zachowywany zgodnie z opisem w sekcji [Zachowanie wyboru](#).

persistence: obiekt HTTP x-msg-persistence , nagłówek jednostki

Ustaw lub zwróć trwałość komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-persistence (trwałość komunikatu)
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	NON_PERSISTENT Komunikat nie jest w stanie przetrwać awariom systemu lub restartów menedżera kolejek. Na przykład składnia <pre>x-msg-persistence: NON_PERSISTENT</pre> PERSISTENT Komunikat przeżyje awarie systemu i restarty menedżera kolejek. Na przykład składnia <pre>x-msg-persistence: PERSISTENT</pre> AS_DESTINATION Ma zastosowanie tylko do produktu POST . Użyj domyślnej trwałości miejsca docelowego określonej przez dostawcę komunikatów. Uwaga: Rozróżnianie wielkości znaków
Wartość domyślna	NON_PERSISTENT

Opis

- W przypadku ustawienia żądania HTTP **POST** ustaw trwałość komunikatu żądania.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-persistence jest ignorowany.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-persistence w komunikacie odpowiedzi HTTP na trwałość komunikatu.

priority: HTTP x-msg-priority entity-header

Ustaw lub zwracaj priorytet komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-priority
Typ nagłówka HTTP	Encja-nagłówek

Typ	Opis
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	LOW Na przykład składnia <pre>x-msg-priority: LOW</pre> MEDIUM Priorytet ten jest równy poziomowi priorytetu IBM MQ równym 4. Na przykład składnia <pre>x-msg-priority: MEDIUM</pre> HIGH Na przykład składnia <pre>x-msg-priority: HIGH</pre> Integer value Łańcuchowa reprezentacja liczby całkowitej z zakresu od 0 do 9, na przykład: <pre>x-msg-priority: 3</pre> AS_DESTINATION Ma zastosowanie tylko do produktu POST . Należy użyć domyślnego priorytetu miejsca docelowego określonego przez dostawcę komunikatów. Uwaga: Rozróżnianie wielkości znaków
Wartość domyślna	MEDIUM

Opis

- W przypadku ustawienia żądania HTTP **POST** ustaw priorytet komunikatu żądania.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-priority jest ignorowany.
- Określone w x-msg-require-headers, ustawia x-msg-priority w komunikacie odpowiedzi HTTP na priorytet komunikatu.

priority: HTTP x-msg-priority entity-header

Ustaw lub zwracaj priorytet komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-priority
Typ nagłówka HTTP	Encja-nagłówków
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers

Typ	Opis
Dozwolone wartości	LOW Na przykład składnia <pre>x-msg-priority: LOW</pre> MEDIUM Priorytet ten jest równy poziomowi priorytetu IBM MQ równym 4. Na przykład składnia <pre>x-msg-priority: MEDIUM</pre> HIGH Na przykład składnia <pre>x-msg-priority: HIGH</pre> Integer value Łańcuchowa reprezentacja liczby całkowitej z zakresu od 0 do 9, na przykład: <pre>x-msg-priority: 3</pre> AS_DESTINATION Ma zastosowanie tylko do produktu POST . Należy użyć domyślnego priorytetu miejsca docelowego określonego przez dostawcę komunikatów. Uwaga: Rozróżnianie wielkości znaków
Wartość domyślna	MEDIUM

Opis

- W przypadku ustawienia żądania HTTP **POST** ustaw priorytet komunikatu żądania.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-priority jest ignorowany.
- Określone w x-msg-require-headers, ustawia x-msg-priority w komunikacie odpowiedzi HTTP na priorytet komunikatu.

replyTo: HTTP x-msg-replyTo entity-header

Ustawia lub zwraca nazwę menedżera kolejek i menedżera kolejek.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-replyTo
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	URI Identyfikator URI punkt z punktem, na przykład:

Typ	Opis
	<pre>x-msg-replyTo: /msg/queue/myReplyQueue x-msg-replyTo: /msg/queue/myReplyQueue@myReplyQueueManager</pre> Uwaga: Rozróżnianie wielkości znaków
Wartość domyślna	MEDIUM

Opis

- W przypadku ustawienia żądania HTTP **POST** ustaw miejsce docelowe komunikatu żądania replyTo .
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-replyTo jest ignorowany.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-replyTo w komunikacie odpowiedzi HTTP na kolejkę odpowiedzi i nazwę menedżera kolejek komunikatu.

Uwaga: Identyfikator URI w odpowiedzi HTTP może zawierać nazwę menedżera kolejek, z którym połączony jest produkt IBM MQ bridge for HTTP .

Serwer: nagłówek odpowiedzi HTTP

Zwraca informacje na temat serwera i protokołu, z którym jest połączony klient.

Typ	Opis
Nazwa nagłówka HTTP	SERVER
Typ nagłówka HTTP	Nagłówek odpowiedzi
Poprawne w komunikacie żądania HTTP	x-msg-require-headers
Wartość zwracana	<pre>WMQ-HTTP/1.1 JEE-Bridge/1.1</pre> lub wersji <pre>Server: Product-token WMQ-HTTP/1.1 JEE-Bridge/1.1</pre>

Opis

- Jeśli most IBM MQ Bridge for HTTP jest wdrażany na serwerze aplikacji, szczegóły IBM MQ bridge for HTTP są dołączane do nagłówka odpowiedzi serwera. Na przykład IBM MQ bridge for HTTP wdrożono w produkcie WebSphere Application Server Community Editiono nazwie Apache-Coyote, co daje odpowiedź:

```
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
```

require-headers: żądanie HTTP x-msg-require-headers -nagłówek

Ustaw, które nagłówki mają zostać zwrócone w komunikacie odpowiedzi HTTP.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-require-headers
Typ nagłówka HTTP	Żądanie-nagłówek
Poprawne w komunikacie żądania HTTP	POST, GET, DELETE

Typ	Opis
Dozwolone wartości	Rozdzielana przecinkami lista nazw nagłówków obiektów: ALL ALL-USR class content-location correlId encoding expiry format msgId NO_require-headers persistence priority replyTo server timestamp usr- <i>property name</i> Na przykład składnia <pre>x-msg-require-headers: msgId</pre> lub <pre>x-msg-require-headers: expiry,correlId,timestamp</pre> Aby zażądać określonej właściwości: <pre>x-msg-require-headers: usr-myCustomProperty</pre> Aby zażądać wszystkich właściwości: <pre>x-msg-require-headers: ALL-USR, ALL</pre>
Wartość domyślna	NO_require-headers

Opis

- W przypadku wartości `x-msg-require-headers` nie jest rozróżniana wielkość liter, z wyjątkiem przypadków stałych `ALL`, `NO_require-headers` i `ALL-USR` oraz zmiennej `property-name`.

timestamp: obiekt HTTP x-msg-timestamp -nagłówek

Zwraca znacznik czasu komunikatu.

Typ	Opis
Nazwa nagłówka HTTP	datownik_x-msg
Typ nagłówka HTTP	Encja-nagłówek

Typ	Opis
Poprawne w komunikacie żądania HTTP	x-msg-require-headers
Wartość zwracana	HTTP-date Data w formacie; dzień, data roku godzina-godzina-zone; na przykład, <pre>Sun, 06 Nov 1994 08:49:37 GMT</pre> Zdefiniowane przez produkt RFC 822i zaktualizowane w produkcie RFC 1123.
Wartość domyślna	Nie dotyczy

Opis

- W przypadku żądania HTTP **POST**, **GET** lub **DELETE** nagłówek x-msg-timestamp jest ignorowany.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-timestamp w komunikacie odpowiedzi HTTP na znacznik czasu komunikatu.

usr: obiekt HTTP x-msg-usr -nagłówek

Ustaw lub zwróć właściwości użytkownika.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-usr
Typ nagłówka HTTP	Encja-nagłówek
Poprawne w komunikacie żądania HTTP	POST , x-msg-require-headers
Dozwolone wartości	Patrz "Składnia" na stronie 1844 ; na przykład, <pre>x-msg-usr: myProp1;5;i1, x-msg-usr: myProp2;"My String";string</pre>
Wartość domyślna	Nie dotyczy

Opis

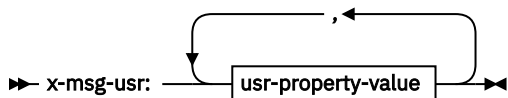
- W przypadku ustawienia w żądaniu HTTP **POST** ustaw właściwości użytkownika komunikatu żądania.
- W przypadku żądania HTTP **GET** lub **DELETE** nagłówek x-msg-usr jest ignorowany.
- Wartość określona w polu x-msg-require-headersustawia wartość x-msg-usr w komunikacie odpowiedzi HTTP na właściwości użytkownika komunikatu.
- W komunikacie można ustawić wiele właściwości. Określ wiele właściwości rozdzielanych przecinkami w jednym nagłówku x-msg-usr lub za pomocą dwóch lub większej liczby oddzielnych instancji nagłówka x-msg-usr .
- Istnieje możliwość żądania zwrócenia określonej właściwości w odpowiedzi na żądanie produktu **GET** lub **DELETE** . Należy określić nazwę właściwości w nagłówku x-msg-require-headers żądania, używając przedrostka usr- . Na przykład składnia

```
x-msg-require-headers: usr-myProp1
```

- Aby zażądać, aby wszystkie właściwości użytkownika zostały zwrócone w odpowiedzi, należy użyć stałej ALL-USR. Na przykład składnia

```
x-msg-require-headers: ALL-USR
```

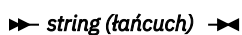
Składnia



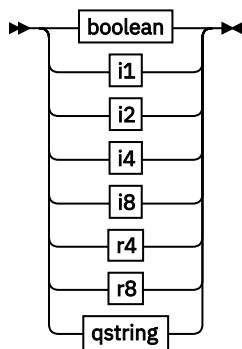
usr-property-value



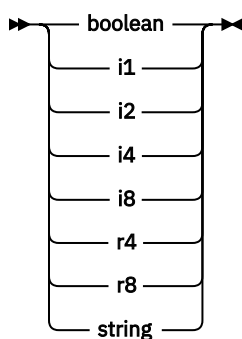
property-name



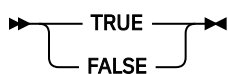
usr-value



usr-type



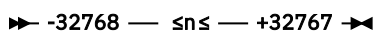
boolean



i1



i2



i4



i8

➤ -9223372036854775808 — ≤n≤ — +92233720368547750807 ➤

r4

➤ -1.4E-45 — ≤n≤ — +3.4028235E38 ➤

r8

➤ -4.9E-324 — ≤n≤ — +1.7976931348623157E308 ➤

qstring

➤ " — *łańcuch* — " ➤

wait: żądanie HTTP x-msg-wait -nagłówek

Ustaw okres oczekiwania na nadejście komunikatu, zanim zostanie zwrócony komunikat odpowiedzi HTTP 504 Gateway Timeout.

Typ	Opis
Nazwa nagłówka HTTP	x-msg-czeka_j
Typ nagłówka HTTP	Żądanie-nagłówek
Poprawne w komunikacie żądania HTTP	GET, DELETE
Dozwolona wartość	NO_WAIT Na przykład składnia <pre>x-msg-wait: NO_WAIT</pre> Integer value Liczba milisekund, przez które IBM MQ bridge for HTTP oczekuje na nadejście komunikatu, na przykład: <pre>x-msg-wait: 8</pre>
Wartość domyślna	NO_WAIT

Opis

- W przypadku żądania HTTP **POST** nagłówek x-msg-wait jest ignorowany.
- W przypadku żądania HTTP **GET** lub **DELETE** parametr x-msg-wait określa czas oczekiwania na przybycie komunikatu, zanim zostanie zwrócona odpowiedź HTTP 504 Gateway Timeout.
- W programie NO_WAIT jest rozróżniana wielkość liter.
- Domyślny maksymalny czas oczekiwania to 35000. Wartość domyślną można zmienić, ustawiając parametr `maximum_wait_time` serwletu. Więcej informacji na ten temat zawiera sekcja [Instalowanie, konfigurowanie i weryfikowanie IBM MQ bridge for HTTP](#).
- Jeśli zostanie ustawiona wartość większa niż `maximum_wait_time`, zamiast niej zostanie użyta wartość `maximum_wait_time`.

Kody powrotu HTTP

Lista kodów powrotu z IBM MQ bridge for HTTP

Program IBM MQ bridge for HTTP zwraca cztery typy błędów:

Błędy serwletu

MQHTTP0001 i MQHTTP0002 są błędami serwletów. Są one rejestrowane, ale nie są zwracane do klienta HTTP.

Pomyślne operacje

Kod statusu HTTP z zakresu od 200 do 299 wskazuje na pomyślne wykonanie operacji.

Błędy klienta

Kod statusu HTTP w zakresie 400-499 wskazuje na błąd klienta. IBM MQ Most dla kodów powrotu HTTP z zakresu MQHTTP40001 - MQHTTP49999 odpowiada błędym klienta.

Błędy serwera

Kod statusu HTTP w zakresie 500-599 wskazuje na błąd klienta. IBM MQ Most dla kodów powrotu HTTP z zakresu MQHTTP50001 - MQHTTP59999 odpowiada błędym serwera.

Jeśli wystąpi błąd serwera, w dzienniku błędów serwera aplikacji zostaną wyświetlone kompletne dane śledzenia stosu. Stos wywołań jest również zwracany do klienta HTTP w odpowiedzi HTTP. Obsłuż stos wywołań w aplikacji klienckiej lub skontaktuj się z administratorem serwera aplikacji, aby rozwiązać ten problem.

Jeśli dane śledzenia stosu zawierają błędy adaptera zasobów, należy zapoznać się z dokumentacją adaptera zasobów.

Alfabetyczna lista kodów powrotu

HTTP 200: OK

Ta klasa kodu statusu wskazuje, że żądanie zostało pomyślnie odebrane, zrozumiane i zaakceptowane.

Kod statusu HTTP

200 OK

HTTP 204: Brak treści

Wysłane następujące po pomyślnym wysłaniu HTTP **GET** lub **DELETE** i x-msg-range: 0 zostały wysłane w żądaniu.

Kod statusu HTTP

204 No Content

MQHTTP0001: Nie określono fabryki połączeń w kontekście serwletu.

Błąd serwletu

Wyjaśnienie

Błąd serwletu

Kod statusu HTTP

Brak

Odpowiedź programisty

Jeśli te błędy są rejestrowane, jest specyficzne dla serwera aplikacji. Zapoznaj się z dokumentacją serwera aplikacji.

MQHTTP0002: Nie można uzyskać menedżera połączeń dla produktu *queueOrTopic* przy użyciu nazwy JNDI serwera *jndiNameTried*.

Błąd serwletu

Wyjaśnienie

Błąd serwletu

Kod statusu HTTP

Brak

Odpowiedź programisty

Jeśli te błędy są rejestrowane, jest specyficzne dla serwera aplikacji. Zapoznaj się z dokumentacją serwera aplikacji.

MQHTTP40001: Zastrzeżone

Zarezerwowane

Kod statusu HTTP

400 Bad Request

MQHTTP40002: Identyfikator URI nie jest poprawny dla transportu produktu IBM MQ dla protokołu HTTP.

Identyfikator URI określony w żądaniu HTTP jest niepoprawny.

Wyjaśnienie

Identyfikator URI określony w żądaniu HTTP jest niepoprawny.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Upewnij się, że format i składnia podanego identyfikatora URI są poprawne.

MQHTTP40003: Identyfikator URI nie jest poprawny. @qmgr jest poprawne tylko dla testu POST

Opcja identyfikatora URI @qmgr została określona w identyfikatorze URI dla żądania HTTP, które nie jest żądaniem **POST**.

Wyjaśnienie

Opcja identyfikatora URI @qmgr została określona w identyfikatorze URI dla żądania HTTP, które nie jest żądaniem **POST**.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Jeśli próbujesz umieścić komunikat za pomocą komendy **POST**, zmień żądanie HTTP na żądanie **POST**. Jeśli podejmowana jest próba pobrania komunikatu za pomocą komend **DELETE** lub **GET**, należy usunąć element @qmgr z identyfikatora URI.

MQHTTP40004: Określono niepoprawny typ Content-Type.

Pole nagłówka Content-Type określone w żądaniu **POST** nie jest zgodne z wartością nagłówka x-msg-class.

Wyjaśnienie

Pole nagłówka Content-Type określone w żądaniu **POST** nie jest zgodne z wartością nagłówka x-msg-class.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Zmień wartość pola nagłówka Content-Type na jeden, który jest obsługiwany. Nagłówek Content-Type musi być zgodny z określonym polem nagłówka x-msg-class.

MQHTTP40005: Błędna wartość nagłówka komunikatu

Podano obsługiwane pole nagłówka z wartością, która nie jest poprawna dla podanego żądania.

Wyjaśnienie

Podano obsługiwane pole nagłówka z wartością, która nie jest poprawna dla podanego żądania.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Zmień wartość określoną dla danego pola nagłówka na poprawną wartość. Sprawdź wielkość liter podanej wartości, ponieważ niektóre pola nagłówka zawierają wartości, w których rozróżniana jest wielkość liter.

MQHTTP40006: *Header_name* nie jest poprawnym nagłówkiem żądania

Nagłówek, który jest poprawny tylko w komunikacie odpowiedzi HTTP, został określony w komunikacie żądania HTTP.

Wyjaśnienie

Nagłówek, który jest poprawny tylko w komunikacie odpowiedzi HTTP, został określony w komunikacie żądania HTTP.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Usuń wszystkie nagłówki z żądania HTTP, które są poprawne tylko w odpowiedzi HTTP; na przykład: x-msg-timestamp.

MQHTTP40007: *Header_name* jest poprawny tylko w dniu ...

Nagłówek został określony w żądaniu HTTP, ale pole nagłówka nie jest poprawne dla danego czasownika żądania.

Wyjaśnienie

Nagłówek został określony w żądaniu HTTP, ale pole nagłówka nie jest poprawne dla danego czasownika żądania.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Usuń wszystkie nagłówki z żądania HTTP, które nie są poprawne dla danego czasownika żądania. Na przykład wartość `x-msg-encoding` jest poprawna dla żądań HTTP **POST**, ale nie jest poprawna dla żądań HTTP **GET** lub HTTP **DELETE**.

MQHTTP40008: Maksymalna długość *Header_name* to ...

Przekroczono maksymalną długość dla danego pola nagłówka.

Wyjaśnienie

Przekroczono maksymalną długość dla danego pola nagłówka.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Zmień wartość pola nagłówka na wartość, która mieści się w zakresie dozwolonym dla pola nagłówka.

MQHTTP40009: Pole nagłówka *header_field* nie jest poprawne dla ...

Pole nagłówka określone w żądaniu HTTP nie jest obsługiwane przez dostawcę przesyłania komunikatów, z którym połączony jest produkt IBM MQ bridge for HTTP.

Wyjaśnienie

Pole nagłówka określone w żądaniu HTTP nie jest obsługiwane przez dostawcę przesyłania komunikatów, z którym połączony jest produkt IBM MQ bridge for HTTP. Błąd występuje wtedy, gdy używany jest dostawca przesyłania komunikatów, który nie może obsługiwać wszystkich funkcji produktu IBM MQ bridge for HTTP.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Usuń nieobsługiwany nagłówek z żądania HTTP.

MQHTTP40010: Nie można przeanalizować komunikatu o typie Content-Type *content_type*.

Treść żądania HTTP nie jest zgodna z typem Content-Type żądania.

Wyjaśnienie

Treść żądania HTTP nie jest zgodna z typem Content-Type żądania. Wspólna przyczyna jest źle uformowana `application/x-www-form-urlencoded` lub `application/xml` data.

Kod statusu HTTP

400 Bad Request

Odpowiedź programisty

Popraw treść żądania HTTP w taki sposób, aby była ona w poprawnym formacie dla typu Content-Type żądania.

MQHTTP40301: Nie można uzyskać dostępu do ...

Program IBM MQ bridge for HTTP nie mógł uwierzytelnić się w określonym miejscu docelowym.

Wyjaśnienie

Program IBM MQ bridge for HTTP nie mógł uwierzytelnić się w określonym miejscu docelowym.

Kod statusu HTTP

403 Forbidden

Odpowiedź programisty

Zmień właściwości uwierzytelniania miejsca docelowego, tak aby most IBM MQ dla protokołu HTTP był autoryzowany do nawiązywania połączenia z nim. Alternatywnie określ miejsce docelowe, do którego ma zostać nawiązane połączenie z produktem IBM MQ bridge for HTTP .

MQHTTP40302: Użytkownik jest zabroniony od ...

Program IBM MQ bridge for HTTP nie mógł połączyć się z menedżerem kolejek.

Wyjaśnienie

Program IBM MQ bridge for HTTP nie mógł połączyć się z menedżerem kolejek. Konfiguracja zabezpieczeń produktu IBM MQ bridge for HTTP jest niepoprawna.

Kod statusu HTTP

403 Forbidden

Odpowiedź programisty

Zmień konfigurację uwierzytelniania menedżera kolejek w taki sposób, aby most IBM MQ dla protokołu HTTP był autoryzowany do nawiązywania połączenia z nim. Alternatywnie można skonfigurować IBM MQ bridge for HTTP do połączenia z menedżerem kolejek, do którego jest uprawniony do połączenia.

MQHTTP40401: Nie można znaleźć miejsca docelowego *destination_name*

Miejsce docelowe określone w identyfikatorze URI żądania HTTP nie może zostać znalezione przez IBM MQ bridge for HTTP.

Wyjaśnienie

Miejsce docelowe określone w identyfikatorze URI żądania HTTP nie może zostać znalezione przez IBM MQ bridge for HTTP.

Kod statusu HTTP

404 Not found

Odpowiedź programisty

Sprawdź, czy miejsce docelowe określone w identyfikatorze URI żądania HTTP istnieje, lub określ alternatywne miejsce docelowe.

MQHTTP40501: Metoda *method_name* nie jest dozwolona

Metoda określona w żądaniu HTTP nie jest obsługiwana przez serwer IBM MQ bridge for HTTP.

Wyjaśnienie

Metoda określona w żądaniu HTTP nie jest obsługiwana przez serwer IBM MQ bridge for HTTP.

Kod statusu HTTP

405 Method not allowed

Odpowiedź programisty

Zmień metodę określoną w żądaniu HTTP na jedną, która jest obsługiwana przez IBM MQ bridge for HTTP.

MQHTTP41301: wysłany komunikat był zbyt duży dla miejsca docelowego.

Miejsce docelowe określone w identyfikatorze URI żądania POST HTTP nie może akceptować komunikatów, które są tak długo, jak jest to komunikat określony w żądaniu HTTP.

Wyjaśnienie

Miejsce docelowe określone w identyfikatorze URI żądania POST HTTP nie może akceptować komunikatów, które są tak długo, jak jest to komunikat określony w żądaniu HTTP.

Kod statusu HTTP

413 Request entity too large

Odpowiedź programisty

Zmniejsz wielkość komunikatu określonego w żądaniu HTTP. Alternatywnie można określić miejsce docelowe, które może obsługiwać komunikaty o pożądanej długości.

MQHTTP41501: Zestaw znaków typu nośnika nie jest obsługiwany

Zestaw znaków określony w polu nagłówka Content-Type nie jest obsługiwany przez IBM MQ bridge for HTTP.

Wyjaśnienie

Zestaw znaków określony w polu nagłówka Content-Type nie jest obsługiwany przez IBM MQ bridge for HTTP.

Kod statusu HTTP

415 Unsupported media type

Odpowiedź programisty

Zmień zestaw znaków w polu nagłówka Content-Type na taki, który jest obsługiwany przez IBM MQ bridge for HTTP.

MQHTTP41502: Typ nośnika *media-type* nie jest obsługiwany ...

Typ nośnika określony w żądaniu HTTP nie jest obsługiwany przez IBM MQ bridge for HTTP dla podanego czasownika HTTP.

Wyjaśnienie

Typ nośnika określony w żądaniu HTTP nie jest obsługiwany przez IBM MQ bridge for HTTP dla podanego czasownika HTTP.

Kod statusu HTTP

415 Unsupported media type

Odpowiedź programisty

Zmień typ nośnika określony w żądaniu HTTP na taki, który jest obsługiwany przez most IBM MQ Bridge dla protokołu HTTP dla określonego czasownika HTTP.

MQHTTP41503: Typ nośnika *media-type* nie jest obsługiwany ...

Typ nośnika określony w żądaniu HTTP nie jest obsługiwany przez IBM MQ bridge for HTTP dla określonego pola nagłówka `x-msg-class`.

Wyjaśnienie

Typ nośnika określony w żądaniu HTTP nie jest obsługiwany przez IBM MQ bridge for HTTP dla określonego pola nagłówka `x-msg-class`.

Kod statusu HTTP

415 Unsupported media type

Odpowiedź programisty

Zmień typ nośnika określony w żądaniu HTTP na taki, który jest obsługiwany przez most IBM MQ Bridge dla protokołu HTTP dla określonego pola nagłówka `x-msg-class`.

MQHTTP41701: Nagłówek HTTP Expect nie jest obsługiwany

IBM MQ bridge for HTTP nie obsługuje pola nagłówka `Expect`.

Wyjaśnienie

Nagłówek `Expect` został określony w żądaniu HTTP. IBM MQ bridge for HTTP nie obsługuje pola nagłówka `Expect`.

Kod statusu HTTP

417 Expectation failed

Odpowiedź programisty

Usuń nagłówek `Expect` z żądania HTTP.

MQHTTP50001: Wystąpił nieoczekiwany problem ...

Wystąpił błąd w IBM MQ bridge for HTTP.

Wyjaśnienie

Wystąpił błąd w IBM MQ bridge for HTTP.

Kod statusu HTTP

500 Internal server error

Odpowiedź programisty

Skontaktuj się z administratorem systemu IBM MQ Bridge for HTTP.

MQHTTP50201: Wystąpił błąd między serwerem IBM MQ bridge for HTTP i menedżerem kolejek.

Wystąpił błąd między IBM MQ bridge for HTTP i menedżerem kolejek

Wyjaśnienie

Wystąpił błąd między IBM MQ bridge for HTTP i menedżerem kolejek

Kod statusu HTTP

502 Bad Gateway

Odpowiedź programisty

Skontaktuj się z administratorem systemu IBM MQ Bridge for HTTP.

MQHTTP50401: Przekroczono limit czasu pobierania komunikatów

W okresie limitu czasu nie został zwrócony żaden komunikat zgodny z podanymi parametrami żądania w HTTP **GET** lub HTTP **DELETE**.

Wyjaśnienie

W okresie limitu czasu nie został zwrócony żaden komunikat zgodny z podanymi parametrami żądania w HTTP **GET** lub HTTP **DELETE**. Kod powrotu wskazuje, że w dowolnym momencie życia żądania HTTP nie był dostępny żaden odpowiedni komunikat.

Kod statusu HTTP

504 Gateway timeout

Odpowiedź programisty

Jeśli oczekiwano komunikatu, sprawdź pola nagłówka w żądaniu HTTP, takie jak `x-msg-correlId` i `x-msg-msgid`. Sprawdź, czy miejsce docelowe określone w identyfikatorze URI żądania HTTP jest poprawne. Spróbuj wydłużenie czasu oczekiwania żądania HTTP, korzystając z pola nagłówka `x-msg-wait`.

MQHTTP50501: HTTP 1.1 i w górę ...

Protokół HTTP używany w żądaniu HTTP nie jest obsługiwany przez serwer IBM MQ bridge for HTTP.

Wyjaśnienie

Protokół HTTP używany w żądaniu HTTP nie jest obsługiwany przez serwer IBM MQ bridge for HTTP.

Kod statusu HTTP

505 HTTP version not supported

Odpowiedź programisty

Zmień żądanie HTTP tak, aby używało protokołu HTTP V1.1 lub nowszego.

Typy komunikatów i odwzorowania komunikatów dla mostu WebSphere dla protokołu HTTP

Produkt IBM MQ bridge for HTTP obsługuje cztery klasy komunikatów: TEXT, BYTES, STREAM i MAP. Klasy komunikatów są odwzorowywane na typy komunikatów produktu JMS i typ Content-TypeHTTP.

HTTP POST

Typ komunikatu, który dociera do miejsca docelowego, zależy od wartości nagłówka `x-msg-class` lub wartości Content-Type żądania HTTP. Tabela 215 na stronie 1853 przedstawia typ Content-Type HTTP, który odpowiada każdej klasie `x-msg-class`. Albo pole może być użyte do ustawienia typu komunikatu i formatu komunikatu. Jeśli oba pola są ustawione, i są ustawione niespójnie, zwracana jest wartość Bad Request exception (HTTP 400, MQHTTP20004).

Tabela 215. Odwzorowanie między elementami <code>x-msg-class</code> i HTTP Content-Type .	
klasa <code>x-msg</code>	HTTP Content-Type
BYTES	application/octet-stream application/xml

Tabela 215. Odzworowanie między elementami <i>x-msg-class</i> i <i>HTTP Content-Type</i> . (kontynuacja)	
klasa x-msg	HTTP Content-Type
TEXT	text/*
MAP	application/x-www-form-urlencoded application/xml (opcjonalny).
STREAM	application/xml (opcjonalny).

Jeśli typ komunikatu JMS jest ustawiony w nagłówku MQRFH2 , to jest on odwzorowywany zgodnie z Tabela 216 na stronie 1854.

Tabela 216. Odzworowanie między typami komunikatów <i>x-msg-class</i> i JMS .	
klasa x-msg	JMS typ komunikatu
BYTES	jms_bytes
TEXT	jms_text
MAP	jms_map
STREAM	jms_stream

Typ komunikatu JMS jest zawsze ustawiany dla klasy komunikatu MAP lub STREAM. Nie zawsze jest ona ustawiana dla klasy komunikatu BYTES lub TEXT. Jeśli dla żądania ma zostać utworzony MQRFH2 , typ komunikatu JMS jest zawsze ustawiany. W przeciwnym razie, jeśli nie zostanie utworzony żaden MQRFH2 , nie zostanie ustawiony żaden typ komunikatu produktu JMS . MQRFH2 jest tworzony, jeśli we wniosku są ustawione właściwości użytkownika, używając nagłówka *x-msg-usr* .

Jeśli typ komunikatu JMS jest ustawiony, to format komunikatu jest ustawiany na wartość MQFMT_NONE, patrz Tabela 218 na stronie 1855:

Tabela 217. Odzworowanie między formatem komunikatu <i>x-msg-class</i> a formatem komunikatu produktu IBM MQ		
klasa x-msg	Format komunikatu z MQRFH2 obecnym w komunikacie	Format komunikatu z <i>nie</i> MQRFH2 obecnym w komunikacie
BYTES	MQFMT_NONE	MQFMT_NONE
TEXT	MQFMT_NONE	MQFMT_STRING
MAP	MQFMT_NONE	Nie jest możliwe
STREAM	MQFMT_NONE	Nie jest możliwe

HTTP GET lub DELETE

Pobrany typ komunikatu lub format określa wartość nagłówka *x-msg-class* i *Content-Type* odpowiedzi HTTP. Nagłówek *x-msg-class* jest zwracany tylko wtedy, gdy jest wymagany w żądaniu *x-msg-headers* .

Tabela 218 na stronie 1855 opisuje odwzorowania między *x-msg-class* i *Content-Type*, a typem komunikatu pobranym z kolejki lub tematu.

Tabela 218. Odzworowywanie typów komunikatów na *x-msg-class* i *Content-Type*

Format wiadomości	JMS Typ komunikatu	klasa x-msg	Content-Type
Wszystko oprócz MQFMT_STRING	Brak	BYTES	application/octet-stream
MQFMT_STRING	Brak	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

Serializacja klas komunikatów MAP i STREAM

Klasy komunikatów MAP i STREAM są przekształcane do postaci szeregowej z powrotem do klienta w odpowiedzi HTTP w taki sam sposób, jak komunikat jest serializowany do kolejki.

W przypadku bazy danych MAP nazwy XML, typy i triplety wartości są kodowane jako:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

STREAM jest podobne do MAP, ale nie ma nazw elementów:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

Uwaga: datatype jest jednym z typów danych zdefiniowanych przy definiowaniu właściwości zdefiniowanych przez użytkownika i wymienionych w [“usr: obiekt HTTP x-msg-usr -nagłówek”](#) na stronie 1843. Atrybut dt="string" jest pomijany w przypadku elementów łańcuchowych, ponieważ domyślnym typem danych jest string.

Format identyfikatora URI

Identyfikatory URI przechwycone przez produkt IBM MQ bridge for HTTP.

Składnia

➤ http: — // — nazwa_hosta — : — port — Ścieżka ➤

Ścieżka

➤ / — contextRoot — / — msg/ — queue/ — queueName — @ — NazwaMgr — / ➤

topic/ — topicName

Uwaga:

1. Jeśli używany jest znak zapytania (?), należy go zastąpić znakiem zapytania %3f. Na przykład wartość `orange?topic` powinna być określona jako `orange%3ftopic`.
2. `@qMgrNazwa` jest poprawny tylko na serwerze HTTP **POST**

Opis

Wdróż serwet produktu IBM MQ bridge for HTTP na serwerze aplikacji produktu JEE przy użyciu kontekstowego katalogu głównego `contextRoot`. Żądania do

```
http://hostname: port/context_root/msg/queue/queueName @ qMgrName
```

i

```
http://hostname: port/context_root/msg/topic/topicString
```

są przechwytywane przez produkt IBM MQ bridge for HTTP.

Klasy i interfejsy IBM MQ .NET

Klasy i interfejsy IBM MQ .NET są uporządkowane alfabetycznie. Opisywane są właściwości, metody i konstruktory.

Klasa MQAsyncStatus.NET

Użyj programu `MQAsyncStatus`, aby dowiedzieć się, jaki jest status poprzedniego działania MQI. Na przykład sprawdź, czy poprzednie asynchroniczne operacje `put` zostały zakończone powodzeniem. Program `MQAsyncStatus` hermetyzuje funkcje struktury danych produktu MQSTS.

Klasa

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```

- [“Właściwości” na stronie 1856](#)
- [“Konstruktory” na stronie 1857](#)

Właściwości

Test dla `MQException` zgłaszanego podczas pobierania właściwości.

```
public static int CompCode {get;}
```

Kod zakończenia od pierwszego błędu lub ostrzeżenia.

```
public static int Reason {get;}
```

Kod przyczyny z pierwszego błędu lub ostrzeżenia.

```
public static int PutSuccessCount {get;}
```

Liczba pomyślnie zakończonych wywołań asynchronicznych wywołań MQI.

```
public static int PutWarningCount {get;}
```

Liczba wywołań asynchronicznych wywołań MQI, które powiodły się z ostrzeżeniem.

public static int PutFailureCount {get;}

Liczba zakończonych niepowodzeniem asynchronicznych wywołań put MQI.

public static int ObjectType {get;}

Typ obiektu dla pierwszego błędu. Dozwolone są następujące wartości:

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q
- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0, co oznacza, że żaden obiekt nie jest zwracany

public static string ObjectName {get;}

Nazwa obiektu.

public static string ObjectQMgrName {get;}

Nazwa menedżera kolejek obiektów.

public static string ResolvedObjectName {get;}

Rozstrzygnięta nazwa obiektu.

public static string ResolvedObjectQMgrName {get;}

Rozstrzygnięta nazwa menedżera kolejek obiektów.

Konstruktory

public MQAsyncStatus() throws MQException;

Metoda konstruktora, konstruuje obiekt z polami, które zostały zainicjowane do wartości zero lub puste, jeśli jest to właściwe.

Klasa MQAuthenticationInformationRecord.NET

Użyj opcji MQAuthenticationInformationRecord , aby określić informacje na temat elementu uwierzytelniającego, który ma być używany w połączeniu klienta SSL produktu IBM MQ .

MQAuthenticationInformationRecord hermetyzuje rekord informacji uwierzytelniających MQAIR.

Klasa

```
System.Object
└─ IBM.WMQ.MQAuthenticationInformationRecord
```

public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;

- [“Właściwości” na stronie 1857](#)
- [“Konstruktory” na stronie 1858](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public long Version {get; set;}

Numer wersji struktury.

public long AuthInfoType {get; set;}

Typ informacji uwierzytelniających. Ten atrybut musi być ustawiony na jedną z następujących wartości:

- OCSP -Sprawdzanie statusu odwołania certyfikatu jest wykonywane przy użyciu protokołu OCSP.
- CRLLDAP -Sprawdzanie statusu odwołania certyfikatu jest wykonywane przy użyciu list odwołań certyfikatów na serwerach LDAP.

public string AuthInfoConnName {get; set;}

Nazwa DNS lub adres IP hosta, na którym działa serwer LDAP, z opcjonalnym numerem portu. To słowo kluczowe jest wymagane.

public string LDAPPassword {get; set;}

Hasło powiązane z nazwą wyróżniającą użytkownika, który uzyskuje dostęp do serwera LDAP. Ta właściwość ma zastosowanie tylko wtedy, gdy parametr **AuthInfoType** jest ustawiony na wartość CRLLDAP.

public string LDAPUserName {get; set;}

Nazwa wyróżniająca użytkownika, który uzyskuje dostęp do serwera LDAP. Po ustawieniu tej właściwości wartości LDAPUserNameLength i LDAPUserNamePtr są automatycznie ustawiane poprawnie. Ta właściwość ma zastosowanie tylko wtedy, gdy parametr AuthInfoType jest ustawiony na wartość CRLLDAP.

public string OCSPResponderURL {get; set;}

Adres URL, przy użyciu którego można nawiązać połączenie z modułem odpowiadającym OCSP. Ta właściwość ma zastosowanie tylko wtedy, gdy parametr AuthInfoType jest ustawiony na wartość OCSP .

W tym polu rozróżniana jest wielkość liter. Musi on rozpoczynać się od łańcucha http:// w postaci małych liter. W pozostałej części adresu URL może być rozróżniana wielkość liter, w zależności od implementacji serwera OCSP.

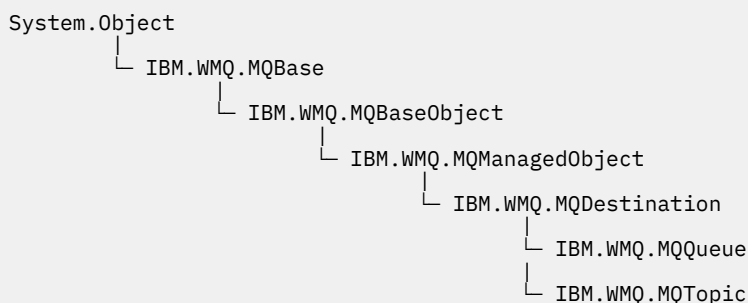
Konstruktory

MQAuthenticationInformationRecord();

Klasa MQDestination.NET

Użyj programu MQDestination , aby uzyskać dostęp do metod, które są wspólne dla produktów MQQueue i MQTopic. MQDestination jest abstrakcyjną klasą bazową i nie można utworzyć jej instancji.

Klasa



public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;

- [“Właściwości” na stronie 1859](#)

- [“Metody” na stronie 1859](#)
- [“Konstruktory” na stronie 1861](#)

Właściwości

Test dla `MQException` zgłaszanego podczas pobierania właściwości.

public DateTime CreationDateTime {get;}

Data i godzina utworzenia kolejki lub tematu. Pierwotnie zawarta w produkcie `MQQueue` właściwość została przeniesiona do podstawowej klasy produktu `MQDestination`.

Nie istnieje wartość domyślna.

public int DestinationType {get;}

Wartość całkowita opisująca typ używanego miejsca docelowego. Zainicjowane z konstruktora klas podrzędnych, `MQQueue` lub `MQTopic`, ta wartość może przyjmować jedną z następujących wartości:

- `MQOT_Q`
- `MQOT_TOPIC`

Nie istnieje wartość domyślna.

Metody

public void Get(MQMessage message);
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);

Zgłasza `MQException`.

Pobiera komunikat z kolejki, jeśli miejscem docelowym jest obiekt `MQQueue`, lub z tematu, jeśli miejscem docelowym jest obiekt `MQTopic`, przy użyciu domyślnej instancji programu `MQGetMessageOptions` do wykonania operacji `get`.

Jeśli operacja pobierania nie powiedzie się, obiekt `MQMessage` nie zostanie zmieniony. Jeśli operacja powiedzie się, deskryptor komunikatu i fragmenty danych komunikatu produktu `MQMessage` zostaną zastąpione przez deskryptor komunikatu i dane komunikatu z komunikatu przychodzącego.

Wszystkie wywołania programu IBM MQ z określonego `MQQueueManager` są synchroniczne. Z tego powodu, jeśli zostanie wykonane oczekiwanie, wszystkie inne wątki korzystające z tego samego serwera `MQQueueManager` są blokowane z kolejnych wywołań programu IBM MQ do czasu zakończenia operacji `Get`. Jeśli wymagane jest jednoczesne uzyskiwanie dostępu do produktu IBM MQ z wielu wątków, każdy wątek musi utworzyć własny obiekt `MQQueueManager`.

message

Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskrypcie komunikatu są parametrami wejściowymi. Ważne jest, aby parametry wejściowe `MessageId` i `CorrelationId` zostały ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny `MQRC_BACKED_OUT` po pomyślnym ponownym nawiązaniu połączenia, dla komunikatów odebranych w ramach produktu `MQGM_SYNCPOINT`.

getMessageOptions

Opcje sterujące działaniem `get`.

Użycie opcji `MQC.MQGMO_CONVERT` może spowodować wystąpienie wyjątku z kodem przyczyny `MQC.MQRC_CONVERTED_STRING_TOO_BIG` podczas przekształcania z jednobajtowych kodów znaków na kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie zostanie określony, użyta zostanie opcja MQGMO_NOWAIT.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQGMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

MaxMsgSize

Największa wiadomość, którą ten obiekt komunikatu ma odebrać. Jeśli komunikat w kolejce jest większy niż ten rozmiar, występuje jedna z dwóch rzeczy:

- Jeśli flaga MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions , to komunikat jest wypełniany możliwie jak największą ilością danych komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_ACCEPTED .
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_FAILED .

Jeśli parametr *MaxMsgSize* nie zostanie określony, zostanie pobrany cały komunikat.

public void Put(MQMessage message);
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

Zgłasza MQException.

Umieszcza komunikat w kolejce, jeśli miejscem docelowym jest obiekt MQQueue , lub opublikuje komunikat w temacie, jeśli miejscem docelowym jest obiekt MQTopic .

Modyfikacje obiektu MQMessage po zakończeniu wywołania Put nie mają wpływu na rzeczywisty komunikat w kolejce IBM MQ ani w temacie publikacji.

Produkt Put aktualizuje właściwości MessageId i CorrelationId obiektu MQMessage i nie powoduje czyszczenia danych komunikatu. Dalsze wywołania programu Put lub Get odnoszą się do zaktualizowanych informacji w obiekcie MQMessage . Na przykład w następującym fragmencie kodu pierwszy komunikat zawiera a i drugi ab.

```
msg.WriteString("a");  
q.Put(msg,pmo);  
msg.WriteString("b");  
q.Put(msg,pmo);
```

message

Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat, który ma zostać wysłany. W wyniku tej metody deskryptor komunikatu może zostać zmieniony. Wartości w deskrytorze komunikatu natychmiast po zakończeniu tej metody to wartości, które zostały umieszczone w kolejce lub opublikowane w temacie.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania Put w trwałym komunikacie, a ponowne nawiązanie połączenia powiodło się.
- MQRC_NONE , jeśli połączenie jest pomyślne podczas uruchamiania wywołania Put w nietrwałym komunikacie (patrz sekcja [Odtwarzanie aplikacji](#)).

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions .

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: Aby uprościć i uzyskać wydajność, należy użyć obiektu `MQQueueManager.Put`, aby umieścić pojedynczy komunikat w kolejce. W tym celu należy mieć dla niego obiekt `MQQueue`.

Konstruktory

`MQDestination` jest abstrakcyjną klasą bazową i nie można utworzyć jej instancji. Dostęp do miejsc docelowych za pomocą konstruktorów `MQQueue` i `MQTopic` lub za pomocą produktów `MQQueueManager.AccessQueue` i `MQQueueManager.AccessTopic` methods.

Klasa `MQEnvironment.NET`

Produkt `MQEnvironment` służy do sterowania sposobem wywołania konstruktora `MQQueueManager` i wybierania połączenia z produktem IBM MQ MQI client. Klasa `MQEnvironment` zawiera właściwości, które sterują zachowaniem IBM MQ.

Klasa

```
System.Object
└─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```

- [“Właściwości-tylko klient” na stronie 1861](#)
- [“Właściwości” na stronie 1862](#)
- [“Konstruktory” na stronie 1863](#)

Właściwości-tylko klient

Test dla `MQException` zgłaszanego podczas pobierania właściwości.

```
public static int CertificateValPolicy {get; set;}
```

Określa, która strategia sprawdzania poprawności certyfikatu SSL/TLS jest używana do sprawdzania poprawności certyfikatów cyfrowych odebranych ze zdalnych systemów partnerskich. Poprawne wartości:

- `MQC.CERTIFICATE_VALIDATION_POLICY_ANY`
- `MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280`

```
public static ArrayList EncryptionPolicySuiteB {get; set;}
```

Ustaw poziom kryptografii zgodny z pakietem B. Poprawne wartości:

- `MQC.MQ_SUITE_B_NONE` -jest to wartość domyślna.
- `MQC.MQ_SUITE_B_128_BIT`
- `MQC.MQ_SUITE_B_192_BIT`

```
public static string Channel {get; set;}
```

Nazwa kanału, z którym ma zostać nawiązane połączenie z docelowym menedżerem kolejek. Wartość *musi* ustawić właściwość kanału przed utworzeniem instancji `MQQueueManager` w trybie klienta.

```
public static int FipsRequired {get; set;}
```

Podaj `MQC.MQSSL_FIPS_YES`, aby używać tylko algorytmów certyfikowanych przez FIPS, jeśli kryptografia jest przeprowadzana w produkcie IBM MQ. Wartością domyślną jest `MQC.MQSSL_FIPS_NO`.

Jeśli sprzęt szyfrujący jest skonfigurowany, używane moduły szyfrujące są dostarczane przez produkt sprzętowy. W zależności od sprzętu, które są w użyciu, mogą nie być zgodne ze standardem FIPS dla określonego poziomu.

public static String Hostname {get; set;}

Nazwa hosta TCP/IP komputera, na którym znajduje się serwer IBM MQ . Jeśli nazwa hosta nie jest ustawiona i nie ustawiono właściwości przestaniających, w celu nawiązania połączenia z lokalnym menedżerem kolejek używany jest tryb powiązań serwera.

public static int Port {get; set;}

Port, z którym ma zostać nawiązane połączenie. Jest to port, na którym serwer IBM MQ nasłuchuje przychodzących żądań połączeń. Wartością domyślną jest 1414.

public static String SSLCipherSpec {get; set;}

Ustaw wartość parametru SSLCipherSpec na wartość parametru CipherSpec ustawioną na kanale SVRCONN, aby włączyć SSL dla połączenia. Wartością domyślną jest NULL, a dla połączenia nie jest włączona obsługa SSL.

public static String sslPeerName {get; set;}

Wzorzec nazwy wyróżniającej. Jeśli ustawiona jest wartość sslCipherSpec , można użyć tej zmiennej, aby upewnić się, że używany jest właściwy menedżer kolejek. Jeśli zostanie ustawiona wartość null (wartość domyślna), nazwa wyróżniająca menedżera kolejek nie jest wykonywana. Parametr sslPeerName jest ignorowany, jeśli parametr sslCipherSpec ma wartość NULL.

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public static ArrayList HdrCompList {get; set;}

Lista kompresji danych nagłówka

public static int KeyResetCount {get; set;}

Wskazuje liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed renegocjacją klucza tajnego.

public static ArrayList MQAIRArray {get; set;}

Tablica obiektów MQAuthenticationInformationRecord .

public static ArrayList MsgCompList {get; set;}

Lista kompresji danych komunikatu

public static String Password {get; set;}

Hasło, które ma zostać uwierzytelnione. Hasło, do którego odwołuje się struktura MQCSP , zostanie zapełnione przez ustawienie tej właściwości Hasło.

public static String ReceiveExit {get; set;}

Wyjście odbierania umożliwia sprawdzenie i zmianę danych odebranych z menedżera kolejek. Jest on zwykle używany z odpowiednim wyjściem wysyłania w menedżerze kolejek. Jeśli parametr ReceiveExit jest ustawiony na wartość null, nie jest wywoływane żadne wyjście odbierania.

public static String ReceiveUserData {get; set;}

Dane użytkownika powiązane z wyjściem odbierania. Ograniczona do 32 znaków.

public static String SecurityExit {get; set;}

Wyjście zabezpieczeń umożliwia dostosowanie przepływów zabezpieczeń, które występują w przypadku próby nawiązania połączenia z menedżerem kolejek. Jeśli parametr SecurityExit jest ustawiony na wartość null, nie jest wywoływane żadne wyjście zabezpieczeń.

public static String SecurityUserData {get; set;}

Dane użytkownika powiązane z wyjściem zabezpieczeń. Ograniczona do 32 znaków.

public static String SendExit {get; set;}

Wyjście wysyłania umożliwia sprawdzenie lub zmianę danych wysyłanych do menedżera kolejek. Jest on zwykle używany z odpowiednim wyjściem odbierania w menedżerze kolejek. Jeśli parametr SendExit jest ustawiony na wartość null, nie jest wywoływane żadne wyjście wysyłania.

public static string SendUserData {get; set;}

Dane użytkownika powiązane z wyjściem wysyłania. Ograniczona do 32 znaków.

public static string SharingConversations {get; set;}

Pole SharingConversations jest używane w połączeniach z aplikacją produktu .NET , gdy te aplikacje nie korzystają z tabeli definicji kanału klienta (CCDT).

Opcja SharingConversations określa maksymalną liczbę konwersacji, które mogą być współużytkowane przez gniazdo powiązane z tym połączeniem.

Wartość 0 oznacza, że kanał działa tak, jak przed programem IBM WebSphere MQ 7.0, w odniesieniu do współużytkowania konwersacji, odczytu z wyprzedzeniem i pulsu.

To pole jest przekazywane w tabeli mieszającej właściwości jako SHARING_CONVERSATIONS_PROPERTY podczas tworzenia instancji menedżera kolejek produktu IBM MQ .

Jeśli opcja SharingConversations nie zostanie określona, zostanie użyta wartość domyślna 10.

public static string SSLCryptoHardware {get; set;}

Ustawia nazwę łańcucha parametru wymaganego do skonfigurowania sprzętu szyfrującego, który jest obecny w systemie. Opcja SSLCryptoHardware jest ignorowana, jeśli specyfikacja sslCipherSpec ma wartość NULL.

public static string SSLKeyRepository {get; set;}

Ustaw pełną nazwę pliku repozytorium kluczy.

Jeśli parametr SSLKeyRepository jest ustawiony na wartość null (wartość domyślna), do znalezienia repozytorium kluczy jest używana zmienna środowiskowa certyfikatu MQSSLKEYR . Opcja SSLCryptoHardware jest ignorowana, jeśli specyfikacja sslCipherSpec ma wartość NULL.

Uwaga: Rozszerzenie .kdb jest obowiązkową częścią nazwy pliku, ale nie jest częścią wartości parametru. Podany katalog musi istnieć. Program IBM MQ tworzy plik po raz pierwszy, gdy uzyskuje dostęp do nowego repozytorium kluczy, chyba że plik już istnieje.

public static string UserId {get; set;}

Identyfikator użytkownika, który ma zostać uwierzytelniony. Identyfikator użytkownika, do którego odwołuje się struktura MQCSP , zostanie wypełniony przez ustawienie UserId. Uwierzytelnianie UserId przy użyciu wyjścia funkcji API lub zabezpieczeń.

Konstruktory

public MQEnvironment()

Klasa MQException.NET

Użyj komendy MQException , aby znaleźć kod zakończenia i przyczyny niepowodzenia funkcji IBM MQ . MQException jest zgłaszany za każdym razem, gdy wystąpi błąd IBM MQ .

Klasa

```
System.Object
├── System.Exception
│   └── System.ApplicationException
│       └── IBM.WMQ.MQException
```

public class IBM.WMQ.MQException extends System.ApplicationException;

- [“Właściwości” na stronie 1864](#)

- [“Konstruktory” na stronie 1864](#)

Właściwości

public int CompletionCode {get; set;}

Kod zakończenia IBM MQ powiązany z błędem. Możliwe wartości:

- MQException.MQCC_OK
- MQException.MQCC_WARNING
- MQException.MQCC_FAILED

public int ReasonCode {get; set;}

IBM MQ kod przyczyny opisujący błąd.

Konstruktory

public MQException(int completionCode, int reasonCode)

completionCode

Kod zakończenia IBM MQ .

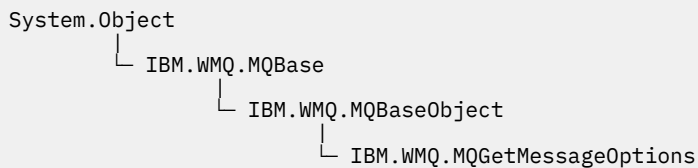
reasonCode

Kod zakończenia IBM MQ .

Klasa MQGetMessageOptions.NET

Użyj opcji MQGetMessageOptions , aby określić, w jaki sposób pobierane są komunikaty. Modyfikuje on działanie produktu MQDestination.Get.

Klasa



```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- [“Właściwości” na stronie 1864](#)
- [“Konstruktory” na stronie 1867](#)

Właściwości

Uwaga: Zachowanie niektórych opcji dostępnych w tej klasie zależy od środowiska, w którym są używane. Te elementy są oznaczone gwiazdką *.

Test dla MQException zgłaszanego podczas pobierania właściwości.

public int GroupStatus {get;}*

GroupStatus wskazuje, czy wczytany komunikat znajduje się w grupie, a jeśli jest ostatnim w grupie. Dozwolone są następujące wartości:

MQC.MQGS_LAST_MSG_IN_GROUP

Komunikat jest ostatnim lub jedynym komunikatem w grupie.

MQC.MQGS_MSG_IN_GROUP

Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

MQC.MQGS_NOT_IN_GROUP

Komunikat nie znajduje się w grupie.

public int MatchOptions {get; set;}*

MatchOptions określa sposób, w jaki zostanie wybrany komunikat. Można ustawić następujące opcje zgodności:

MQC.MQMO_MATCH_CORREL_ID

Identyfikator korelacji do dopasowania.

MQC.MQMO_MATCH_GROUP_ID

Identyfikator grupy do dopasowania.

MQC.MQMO_MATCH_MSG_ID

Identyfikator komunikatu, który ma zostać dopasowany.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER

Dopasuj numer kolejny komunikatu.

MQC.MQMO_NONE

Nie jest wymagane żadne dopasowanie.

public int Options {get; set;}*

Opcje sterują działaniem programu MQQueue.get. Możliwe jest określenie dowolnej z poniższych wartości. Jeśli wymagana jest więcej niż jedna opcja, wartości można dodawać lub łączyć za pomocą operatora bitowego OR.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG

Zezwalaj na obcinanie danych komunikatu.

MQC.MQGMO_ALL_MSGS_AVAILABLE*

Pobieranie komunikatów z grupy tylko wtedy, gdy wszystkie komunikaty w grupie są dostępne.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*

Pobieranie segmentów komunikatu logicznego tylko wtedy, gdy wszystkie segmenty w grupie są dostępne.

MQC.MQGMO_BROWSE_FIRST

Odszukaj od początku kolejki.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*

Przeglądaj kursor pod kurorem przeglądania.

MQC.MQGMO_BROWSE_NEXT

Przeglądaj z bieżącej pozycji w kolejce.

MQC.MQGMO_COMPLETE_MSG*

Pobieranie tylko pełnych komunikatów logicznych.

MQC.MQGMO_CONVERT

Zażądaj konwersji danych aplikacji, aby były one zgodne z atrybutami CharSet i Encoding serwera MQMessage, zanim dane zostaną skopiowane do buforu komunikatów. Ponieważ konwersja danych jest stosowana również w przypadku pobierania danych z buforu komunikatów, aplikacje nie ustawiają tej opcji.

Użycie tej opcji może spowodować problemy podczas konwersji z jednobajtowych zestawów znaków na dwubajtowe zestawy znaków. Zamiast tego należy wykonać konwersję przy użyciu metod readString, readLinei writeString po dostarczeniu komunikatu.

MQC.MQGMO_FAIL_IF QUIESCING

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

MQC.MQGMO_LOCK*

Zablokuj przejrany komunikat.

MQC.MQGMO_LOGICAL_ORDER*

Zwracane są komunikaty w grupach i segmentach komunikatów logicznych w porządku logicznym.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQGMO_LOGICAL_ORDER, kod przyczyny MQRC_RECONNECT_INCOMPATIBLE jest zwracany do aplikacji.

MQC.MQGMO_MARK_SKIP_BACKOUT*

Zezwalaj na wycofywanie jednostki pracy bez ponownego wprowadzenia komunikatu w kolejce.

MQC.MQGMO_MSG_UNDER_CURSOR

Pobierz komunikat pod kursorem przeglądania.

MQC.MQGMO_NONE

Nie określono żadnych innych opcji. Wszystkie opcje przyjmują wartości domyślne.

MQC.MQGMO_NO_PROPERTIES

Nie są pobierane żadne właściwości komunikatu, z wyjątkiem właściwości zawartych w deskrypcji komunikatu (lub rozszerzeniu).

MQC.MQGMO_NO_SYNCPOINT

Pobierz komunikat bez elementu sterującego punktu synchronizacji.

MQC.MQGMO_NO_WAIT

Zwróć natychmiast, jeśli nie ma odpowiedniego komunikatu.

MQC.MQGMO_PROPERTIES_AS_Q_DEF

Pobieranie właściwości komunikatu zgodnie z definicją atrybutu PropertyControl produktu MQQueue. Dostęp do właściwości komunikatu w deskrypcji komunikatu lub rozszerzeniu nie ma wpływu na atrybut PropertyControl.

MQC.MQGMO_PROPERTIES_COMPATIBILITY

Pobieranie właściwości komunikatu z przedrostkiem mcd, jms, uslub mqext, w nagłówkach MQRFH2. Pozostałe właściwości komunikatu, z wyjątkiem właściwości zawartych w deskrypcji komunikatu lub rozszerzeniu, są usuwane.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2

Pobieranie właściwości komunikatu, z wyjątkiem właściwości zawartych w deskrypcji komunikatu lub rozszerzeniu, w nagłówkach MQRFH2. Należy użyć produktu MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 w aplikacjach, które oczekują na pobranie właściwości, ale nie można ich zmienić w celu użycia uchwytów komunikatów.

MQC.MQGMO_PROPERTIES_IN_HANDLE

Pobierz właściwości komunikatu przy użyciu elementu MsgHandle.

MQC.MQGMO_SYNCPOINT

Pobierz komunikat pod kontrolą punktu synchronizacji. Komunikat jest oznaczony jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzana. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy jest wycofana.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*

Jeśli komunikat jest trwały, pobierz komunikat z elementem sterującym punktu synchronizacji.

MQC.MQGMO_UNLOCK*

Odblokuj poprzednio zablokowany komunikat.

MQC.MQGMO_WAIT

Poczekaj na przybycie komunikatu.

public string ResolvedQueueName {get;}

Menadżer kolejek ustawia nazwę ResolvedQueueName na lokalną nazwę kolejki, z której został pobrany komunikat. Opcja ResolvedQueueName różni się od nazwy używanej do otwarcia kolejki, jeśli kolejka aliasowa lub kolejka modelowa została otwarta.

public char Segmentation {get;}*

Segmentacja wskazuje, czy możliwa jest segmentacja dla pobranego komunikatu. Dozwolone są następujące wartości:

MQC.MQSEG_INHIBITED

Nie zezwalaj na segmentację.

MQC.MQSEG_ALLOWED

Zezwalaj na segmentację

public byte SegmentStatus {get;}*

SegmentStatus to pole wyjściowe, które wskazuje, czy pobrany komunikat jest segmentem komunikatu logicznego. Jeśli komunikat jest segmentem, flaga wskazuje, czy jest to ostatni segment. Dozwolone są następujące wartości:

MQC.MQSS_LAST_SEGMENT

Komunikat jest ostatnim lub jedynym segmentem komunikatu logicznego.

MQC.MQSS_NOT_A_SEGMENT

Komunikat nie jest segmentem.

MQC.MQSS_SEGMENT

Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

public int WaitInterval {get; set;}

WaitInterval to maksymalny czas (w milisekundach), przez jaki wywołanie MQQueue.get oczekuje na nadejście odpowiedniego komunikatu. Opcji WaitInterval należy używać z produktem MQC.MQGMO_WAIT. Ustaw wartość MQC.MQWI_UNLIMITED, aby oczekiwać nieograniczonego czasu na komunikat.

Konstruktory

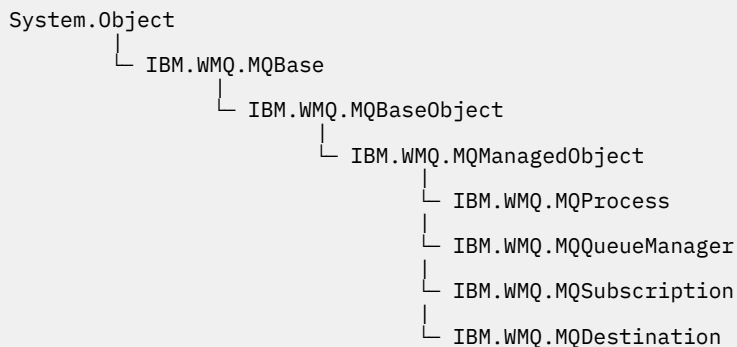
public MQGetMessageOptions()

Skonstruuj nowy obiekt MQGetMessageOptions z wartością Options ustawioną na wartość MQC.MQGMO_NO_WAIT, WaitInterval ustawioną na zero, a parametr ResolvedQueueName ma wartość pustą.

Klasa MQManagedObject.NET

MQManagedObject umożliwia sprawdzenie i ustawianie atrybutów produktów MQDestination, MQProcess, MQQueueManager i MQSubscription. MQManagedObject jest nadklasą tych klas.

Klasy



```
public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;
```

- [“Właściwości” na stronie 1867](#)
- [“Metody” na stronie 1868](#)
- [“Konstruktory” na stronie 1869](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public string AlternateUserId {get; set;}

Alternatywny ID użytkownika, jeśli istnieje, ustawiany podczas otwierania zasobu. Opcja AlternateUserId.set jest ignorowana, jeśli jest wydawana dla obiektu, który jest otwarty. Obiekt AlternateUserId nie jest poprawny dla subskrypcji.

public int CloseOptions {get; set;}

Ustaw ten atrybut, aby kontrolować sposób zamykania zasobu. Wartością domyślną jest MQC.MQCO_NONE. MQC.MQCO_NONE jest jedyną dopuszczalną wartością dla wszystkich zasobów innych niż trwałe kolejki dynamiczne, tymczasowe kolejki dynamiczne, subskrypcje i tematy, do których dostęp jest uzyskiwany przez obiekty, które je utworzyły.

W przypadku kolejek i tematów dopuszczalne są następujące wartości dodatkowe:

MQC.MQCO_DELETE

Usuń kolejkę, jeśli nie ma żadnych komunikatów.

MQC.MQCO_DELETE_PURGE

Usuń kolejkę, wyczyszczając na niej wszystkie komunikaty.

MQC.MQCO_QUIESCE

Zażądaj zamknięcia kolejki, jeśli zostanie wyświetlone ostrzeżenie, jeśli zostaną wyświetlone jakiegokolwiek komunikaty (co pozwoli na pobranie ich przed ostatecznym zamknięciem).

W przypadku subskrypcji dopuszczalne są następujące wartości dodatkowe:

MQC.MQCO_KEEP_SUB

Subskrypcja nie została usunięta. Ta opcja jest poprawna tylko wtedy, gdy oryginalna subskrypcja jest trwałą. MQC.MQCO_KEEP_SUB jest wartością domyślną dla trwałego tematu.

MQC.MQCO_REMOVE_SUB

Subskrypcja została usunięta. MQC.MQCO_REMOVE_SUB jest wartością domyślną dla nietrwałego tematu niezarządzanego.

MQC.MQCO_PURGE_SUB

Subskrypcja została usunięta. MQC.MQCO_PURGE_SUB jest wartością domyślną dla tematu, który nie jest trwały.

public MQQueueManager ConnectionReference {get;}

Menedżer kolejek, do którego należy ten zasób.

public string MQDescription {get;}

Opis zasobu przechowanego przez menedżera kolejek. MQDescription zwraca pusty łańcuch dla subskrypcji i tematów.

public boolean IsOpen {get;}

Wskazuje, czy zasób jest aktualnie otwarty.

public string Name {get;}

Nazwa zasobu. Nazwa jest dostarczona w metodzie dostępu lub jest przydzielona przez menedżer kolejek dla kolejki dynamicznej.

public int OpenOptions {get; set;}

OpenOptions są ustawiane, gdy obiekt IBM MQ jest otwarty. Metoda OpenOptions.set jest ignorowana i nie powoduje wystąpienia błędu. Subskrypcje nie mają OpenOptions.

Metody

public virtual void Close();

Zgłasza MQException.

Zamyka obiekt. Po wywołaniu programu Closures są dozwolone żadne dalsze operacje dotyczące tego zasobu. Aby zmienić sposób działania metody Close, należy ustawić atrybut closeOptions.

public string GetAttributeString(int selector, int length);

Zgłasza MQException.

Pobiera łańcuch atrybutu.

selector

Liczba całkowita wskazująca, który atrybut jest odpytywany.

length

Liczba całkowita określająca długość wymaganego łańcucha.

public void Inquire(int[] selectors, int[] intAttrs, byte[] charAttrs);

Zgłasza `MQException`.

Zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty kolejki, procesu lub menedżera kolejek. Atrybuty, które mają być odpytywane, są określone w tablicy selektorów.

Uwaga: Wiele z tych atrybutów można odpytywać za pomocą metod `Get` zdefiniowanych w `MQManagedObject`, `MQQueue` i `MQQueueManager`.

selectors

Tablica liczb całkowitych identyfikująca atrybuty z wartościami, które mają zostać zapytane.

intAttrs

Tablica, w której zwracane są wartości atrybutów całkowitych. Wartości atrybutów całkowitych są zwracane w tej samej kolejności, w jakiej znajdują się selektory atrybutów w postaci liczby całkowitej w tablicy selektorów.

charAttrs

Bufor, w którym zwracane są atrybuty znakowe, konkatelowane. Atrybuty znaków są zwracane w tej samej kolejności, co selektory atrybutów znakowych w tablicy selektorów. Długość każdego łańcucha atrybutu jest stała dla każdego atrybutu.

public void Set(int[] selectors, int[] intAttrs, byte[] charAttrs);

Zgłasza `MQException`.

Ustawia atrybuty zdefiniowane w wektorze selektorów. Atrybuty, które mają zostać ustawione, są określone w tablicy selektorów.

selectors

Tablica liczb całkowitych identyfikująca atrybuty z wartościami, które mają zostać ustawione.

intAttrs

Tablica wartości atrybutów całkowitoliczbowych, które mają zostać ustawione. Wartości te muszą być w tej samej kolejności, w jakiej znajdują się selektory atrybutów w postaci liczby całkowitej w tablicy selektorów.

charAttrs

Bufor, w którym atrybuty znakowe, które mają być ustawione, są konkatelowane. Wartości te muszą być w tej samej kolejności, w jakiej znajdują się selektory atrybutów znakowych w tablicy selektorów. Długość każdego atrybutu znaku jest stała.

public void SetAttributeString(int selector, string value, int length);

Zgłasza `MQException`.

Ustawia łańcuch atrybutu.

selector

Liczba całkowita wskazująca, który atrybut jest ustawiany.

value

Łańcuch, który ma zostać ustawiony jako wartość atrybutu.

length

Liczba całkowita określająca długość wymaganego łańcucha.

Konstruktory

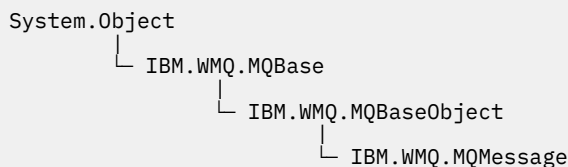
protected MQManagedObject()

Metoda konstruktora. Ten obiekt jest abstrakcyjną klasą bazową, której nie można utworzyć samodzielnie.

Klasa MQMessage.NET

Użyj opcji MQMessage , aby uzyskać dostęp do deskryptora komunikatu i danych dla komunikatu IBM MQ . MQMessage hermetyzuje komunikat IBM MQ .

Klasa



```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Utwórz obiekt MQMessage , a następnie użyj metod Read i Write w celu przesłania danych między komunikatem i innymi obiektami w aplikacji. Wysyłanie i odbieranie obiektów MQMessage przy użyciu metod Put i Get klas MQDestination, MQQueue i MQTopic .

Pobierz i ustaw właściwości deskryptora komunikatu przy użyciu właściwości produktu MQMessage. Ustaw właściwości rozszerzonego komunikatu i pobierz je za pomocą metod SetProperty i GetProperty .

- [“Właściwości” na stronie 1870](#)
- [“Metody komunikatów Read i Write” na stronie 1876](#)
- [“Metody buforowania” na stronie 1879](#)
- [“Metody właściwości” na stronie 1880](#)
- [“Konstruktory” na stronie 1882](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

```
public string AccountingToken {get; set;}
```

Część kontekstu tożsamości komunikatu. Pomaga ona aplikacji obciążania za pracę wykonanego w wyniku komunikatu. Wartością domyślną jest MQC . MQACT_NONE.

```
public string ApplicationIdData {get; set;}
```

Część kontekstu tożsamości komunikatu. ApplicationIdData to informacje, które są definiowane przez pakiet aplikacji i mogą być używane w celu udostępnienia dodatkowych informacji na temat komunikatu lub jego inicjatora. Wartością domyślną jest " " .

```
public string ApplicationOriginData {get; set;}
```

Informacje zdefiniowane przez aplikację, które mogą być używane w celu udostępnienia dodatkowych informacji o pochodzeniu komunikatu. Wartością domyślną jest " " .

```
public int BackoutCount {get;}
```

Liczba przypadków, w których komunikat został wcześniej zwrócony i wycofany przez wywołanie MQQueue . Get w ramach jednostki pracy. Wartość domyślna to zero.

```
public int CharacterSet {get; set;}
```

Identyfikator kodowanego zestawu znaków danych znakowych w komunikacie.

Aby zidentyfikować zestaw znaków danych znakowych w komunikacie, należy ustawić parametr CharacterSet . Pobierz CharacterSet , aby dowiedzieć się, jaki zestaw znaków został użyty do kodowania danych znakowych w komunikacie.

Aplikacje produktu .NET zawsze działają w kodzie Unicode, podczas gdy w innych środowiskach aplikacje działają w tym samym zestawie znaków, w którym działa menedżer kolejek.

Metody `ReadString` i `ReadLine` przekształcają dane znakowe w komunikacie na Unicode dla użytkownika.

Metoda `WriteString` przekształca kod Unicode na zestaw znaków zakodowany w `CharacterSet`. Jeśli właściwość `CharacterSet` jest ustawiona na wartość domyślną, `MQC.MQCCSI_Q_MGR`, która wynosi 0, konwersja nie ma miejsca, a parametr `CharacterSet` jest ustawiony na wartość 1200. Jeśli wartość parametru `CharacterSet` zostanie ustawiona na inną wartość, program `WriteString` przekształci z kodu Unicode na wartość alternatywną.

Uwaga: Inne metody odczytu i zapisu nie korzystają z elementu `CharacterSet`.

- Produkty `ReadChar` i `WriteChar` odczytane i zapisują znak Unicode do i z buforu komunikatów bez konwersji.
- `ReadUTF` i `WriteUTF` konwertują między łańcuchem Unicode w aplikacji, a łańcuchem UTF-8, poprzedzonym polem o długości 2 bajtów, w buforze komunikatów.
- Metody bajtowe przesyłają bajty między aplikacją a buforem komunikatów bez zmiany.

public byte[] CorrelationId {get; set;}

- W przypadku wywołania `MQQueue.Get` identyfikator korelacji komunikatu, który ma zostać pobrany. Menedżer kolejek zwraca pierwszy komunikat z identyfikatorem komunikatu i identyfikatorem korelacji, który jest zgodny z polami deskryptora komunikatu. Wartość domyślna `MQC.MQCI_NONE` pomaga w dopasowaniu dowolnego identyfikatora korelacji.
- W przypadku wywołania `MQQueue.Put` identyfikator korelacji do ustawienia.

public int DataLength {get;}

Liczba bajtów, które pozostały do odczytu.

public int DataOffset {get; set;}

Bieżąca pozycja kursora w danych komunikatu. Odczyty i zapisy są aktywne w bieżącej pozycji.

public int Encoding {get; set;}

Reprezentacja używana dla wartości liczbowych w danych komunikatu aplikacji. Kodowanie ma zastosowanie do danych binarnych, upakowanych liczb dziesiętnych i zmiennopozycyjnych. Zachowanie metod odczytu i zapisu dla tych formatów liczbowych jest odpowiednio zmieniane. Skonstruuj wartość dla pola kodowania, dodając jedną wartość z każdej z tych trzech sekcji. Alternatywnie można skonstruować wartość łączącą wartości z każdej z trzech sekcji przy użyciu operatora bitowego OR.

1. binarna liczba całkowita

MQC.MQENC_INTEGER_NORMAL

Big-endian liczb całkowitych.

MQC.MQENC_INTEGER_REVERSED

Little-endian liczb całkowitych, zgodnie z używanym w architekturze Intel.

2. Zapakowane-dziesiętne

MQC.MQENC_DECIMAL_NORMAL

Big-endian packed-decimal, używany przez produkt z/OS.

MQC.MQENC_DECIMAL_REVERSED

Little-endian packed-decimal.

3. zmiennopozycyjne

MQC.MQENC_FLOAT_IEEE_NORMAL

Big-endian IEEE floats.

MQC.MQENC_FLOAT_IEEE_REVERSED

Little-endian IEEE floats, as used Intel architecture.

MQC.MQENC_FLOAT_S390

z/OS -format zmiennopozycyjny.

Wartość domyślna to:

```
MQC.MQENC_INTEGER_REVERSED |  
MQC.MQENC_DECIMAL_REVERSED |  
MQC.MQENC_FLOAT_IEEE_REVERSED
```

Ustawienie domyślne powoduje, że program `WriteInt` zapisuje liczbę całkowitą little endian, a program `ReadInt` odczyta liczbę całkowitą z little endian. Jeśli zamiast niej zostanie ustawiona flaga `MQC.MQENC_INTEGER_NORMAL`, program `WriteInt` zapisze dużą liczbę całkowitą, a program `ReadInt` odczyta big-endian integer.

Uwaga: Utrata precyzji może wystąpić podczas konwersji z liczb zmiennopozycyjnych IEEE na punkty zmiennopozycyjne w formacie `zSeries`.

public int Expiry {get; set;}

Czas utraty ważności wyrażony w dziesiątych częściach sekundy, ustawiany przez aplikację, która umieszcza komunikat. Po upływie czasu utraty ważności komunikatu kwalifikuje się on do odrzucenia przez menedżer kolejek. Jeśli w komunikacie określono jedną z opcji `MQC.MQRO_EXPIRATION`, raport jest generowany, gdy komunikat jest odrzucany. Wartością domyślną jest `MQC.MQEI_UNLIMITED`, co oznacza, że komunikat nigdy nie traci ważności.

public int Feedback {get; set;}

Użyj opcji Feedback (Opinia) z komunikatem typu `MQC.MQMT_REPORT`, aby wskazać rodzaj raportu. Następujące kody sprzężenia zwrotnego są definiowane przez system:

- `MQC.MQFB_EXPIRATION`
- `MQC.MQFB_COA`
- `MQC.MQFB_COD`
- `MQC.MQFB_QUIT`
- `MQC.MQFB_PAN`
- `MQC.MQFB_NAN`
- `MQC.MQFB_DATA_LENGTH_ZERO`
- `MQC.MQFB_DATA_LENGTH_NEGATIVE`
- `MQC.MQFB_DATA_LENGTH_TOO_BIG`
- `MQC.MQFB_BUFFER_OVERFLOW`
- `MQC.MQFB_LENGTH_OFF_BY_ONE`
- `MQC.MQFB_IIH_ERROR`

Można również użyć wartości informacji zwrotnych zdefiniowanych przez aplikację z zakresu od `MQC.MQFB_APPL_FIRST` do `MQC.MQFB_APPL_LAST`. Wartością domyślną tego pola jest `MQC.MQFB_NONE`, co oznacza, że nie podano żadnych informacji zwrotnych.

public string Format {get; set;}

Nazwa formatu używana przez nadawcę komunikatu w celu wskazania rodzaju danych w komunikacie do odbiorcy. Można użyć własnych nazw formatów, ale nazwy rozpoczynające się od liter MQ mają znaczenie, które są zdefiniowane przez menedżer kolejek. Wbudowane formaty menedżera kolejek to:

MQC.MQFMT_ADMIN

Komunikat żądania/odpowiedzi serwera komend.

MQC.MQFMT_COMMAND_1

Komunikat odpowiedzi komendy typu 1.

MQC.MQFMT_COMMAND_2

Komunikat odpowiedzi komendy typu 2.

MQC.MQFMT_DEAD_LETTER_HEADER

Nagłówek niewysłanych wiadomości.

MQC.MQFMT_EVENT

Komunikat zdarzenia.

MQC.MQFMT_NONE

Brak nazwy formatu.

MQC.MQFMT_PCF

Komunikat zdefiniowany przez użytkownika w formacie komendy programowalnej.

MQC.MQFMT_STRING

Komunikat składający się całkowicie z znaków.

MQC.MQFMT_TRIGGER

komunikat wyzwacza

MQC.MQFMT_XMIT_Q_HEADER

Nagłówek kolejki transmisji.

Wartością domyślną jest MQC.MQFMT_NONE.

public byte[] GroupId {get; set;}

Łańcuch bajtowy identyfikujący grupę komunikatów, do której należy komunikat fizyczny. Wartością domyślną jest MQC.MQGI_NONE.

public int MessageFlags {get; set;}

Flagi sterujące segmentacją i statusem komunikatu.

public byte[] MessageId {get; set;}

W przypadku wywołania MQQueue.Get to pole określa identyfikator komunikatu, który ma zostać pobrany. W normalnych warunkach menedżer kolejek zwraca pierwszy komunikat z identyfikatorem komunikatu i identyfikatorem korelacji, które są zgodne z polami deskryptora komunikatu. Zezwalaj na zgodność z dowolnym identyfikatorem komunikatu przy użyciu wartości specjalnej MQC.MQMI_NONE.

W przypadku wywołania MQQueue.Put to pole określa identyfikator komunikatu, który ma być używany. Jeśli określono wartość MQC.MQMI_NONE, menedżer kolejek generuje unikalny identyfikator komunikatu, gdy komunikat jest umieszczany. Wartość tej zmiennej składowej jest aktualizowana po umieszczeniu w celu wskazania identyfikatora komunikatu, który został użyty. Wartością domyślną jest MQC.MQMI_NONE.

public int MessageLength {get;}

Liczba bajtów danych komunikatu w obiekcie MQMessage.

public int MessageSequenceNumber {get; set;}

Numer kolejny komunikatu logicznego w grupie.

public int MessageType {get; set;}

Wskazuje typ komunikatu. Następujące wartości są obecnie zdefiniowane przez system:

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

Wartości zdefiniowane przez aplikację mogą być również używane, w zakresie od MQC.MQMT_APPL_FIRST do MQC.MQMT_APPL_LAST. Wartością domyślną tego pola jest MQC.MQMT_DATAGRAM.

public int Offset {get; set;}

W posegmentowanym komunikacie przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

public int OriginalLength {get; set;}

Oryginalna długość segmentowanego komunikatu.

public int Persistence {get; set;}

Trwałość komunikatu. Zdefiniowane są następujące wartości:

- MQC.MQPER_NOT_PERSISTENT

Jeśli ta opcja zostanie ustawiona w kliencie z możliwością ponownego połączenia, kod przyczyny MQRC_NONE zostanie zwrócony do aplikacji, gdy połączenie zakończy się pomyślnie.

- MQC.MQPER_PERSISTENT

Jeśli ta opcja zostanie ustawiona w kliencie z możliwością ponownego połączenia, kod przyczyny produktu MQRC_CALL_INTERRUPTED zostanie zwrócony do aplikacji po pomyślnym nawiązaniu połączenia.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

Wartością domyślną jest MQC.MQPER_PERSISTENCE_AS_Q_DEF, która pobiera trwałość komunikatu z domyślnego atrybutu trwałości w kolejce docelowej.

public int Priority {get; set;}

Priorytet komunikatu. Wartość specjalną MQC.MQPRI_PRIORITY_AS_Q_DEF może być również ustawiona w komunikacie wychodzącym. Priorytet dla komunikatu jest następnie przyjmowany z domyślnego atrybutu priorytetu kolejki docelowej. Wartością domyślną jest MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}

Określa, czy sprawdzanie poprawności właściwości ma miejsce, gdy właściwość komunikatu jest ustawiona. Dozwolone są następujące wartości:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

Wartością domyślną jest MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}

Nazwa aplikacji umieszczonej w komunikacie. Wartością domyślną jest "".

public int PutApplicationType {get; set;}

Typ aplikacji, która wstawiła komunikat. PutApplicationTyp może być wartością zdefiniowaną przez system lub zdefiniowaną przez użytkownika. System definiuje następujące wartości:

- MQC.MQAT_AIX
- MQC.MQAT_CICS
- MQC.MQAT_DOS
- MQC.MQAT_IMS
- MQC.MQAT_MVS
- MQC.MQAT_OS2
- MQC.MQAT_OS400
- MQC.MQAT_QMGR
- MQC.MQAT_UNIX
- MQC.MQAT_WINDOWS
- MQC.MQAT_JAVA

Wartością domyślną jest MQC.MQAT_NO_CONTEXT, co oznacza, że w komunikacie nie ma informacji o kontekście.

public DateTime PutDateTime {get; set;}

Data i godzina umieszczenia komunikatu.

public string ReplyToQueueManagerName {get; set;}

Nazwa menedżera kolejek, który ma wysyłać komunikaty odpowiedzi lub raporty. Wartością domyślną jest "", a menedżer kolejek udostępnia nazwę ReplyToQueueManagerName.

public string ReplyToQueueName {get; set;}

Nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania dla komunikatu, wysyła komunikaty MQC.MQMT_REPLY i MQC.MQMT_REPORT. Wartością domyślną parametru ReplyToQueueName jest "".

public int Report {get; set;}

Użyj opcji Report, aby określić opcje dotyczące komunikatów raportu i odpowiedzi:

- Określa, czy raporty są wymagane.
- Określa, czy dane komunikatu aplikacji mają być uwzględniane w raportach.
- W jaki sposób ustawić identyfikatory komunikatów i korelacji w raporcie lub odpowiedzi.

Można zażądać dowolnej kombinacji czterech typów raportów:

- Określ dowolną kombinację czterech typów raportów. Wybranie dowolnej z trzech opcji dla każdego typu raportu, w zależności od tego, czy dane komunikatu aplikacji mają zostać uwzględnione w komunikacie raportu.

1. Potwierdź po przybyciu

- MQC.MQRO_COA
- MQC.MQRO_COA_WITH_DATA
- MQC.MQRO_COA_WITH_FULL_DATA **

2. Potwierdź przy dostarczeniu

- MQC.MQRO_COD
- MQC.MQRO_COD_WITH_DATA
- MQC.MQRO_COD_WITH_FULL_DATA **

3. Wyjątek

- MQC.MQRO_EXCEPTION
- MQC.MQRO_EXCEPTION_WITH_DATA
- MQC.MQRO_EXCEPTION_WITH_FULL_DATA **

4. Termin ważności

- MQC.MQRO_EXPIRATION
- MQC.MQRO_EXPIRATION_WITH_DATA
- MQC.MQRO_EXPIRATION_WITH_FULL_DATA **

Uwaga: Wartości oznaczone znakiem ** na liście nie są obsługiwane przez menedżery kolejek produktu z/OS. Nie należy ich używać, jeśli aplikacja może uzyskać dostęp do menedżera kolejek produktu z/OS, niezależnie od platformy, na której działa aplikacja.

- Określ jedną z poniższych opcji, aby określić sposób generowania identyfikatora komunikatu dla komunikatu lub komunikatu odpowiedzi:
 - MQC.MQRO_NEW_MSG_ID
 - MQC.MQRO_PASS_MSG_ID
- Określ jedną z następujących opcji, aby określić, w jaki sposób identyfikator korelacji komunikatu lub komunikatu odpowiedzi ma być ustawiony:
 - MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID
 - MQC.MQRO_PASS_CORREL_ID
- Określ jedną z następujących opcji, aby sterować rozporządzaniem oryginalnego komunikatu, gdy nie może zostać dostarczony do kolejki docelowej:
 - MQC.MQRO_DEAD_LETTER_Q
 - MQC.MQRO_DISCARD_MSG **
- Jeśli nie zostaną podane żadne opcje raportu, wartością domyślną jest:

```
MQC.MQRO_NEW_MSG_ID |  
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |  
MQC.MQRO_DEAD_LETTER_Q
```

- Aby zażądać, aby aplikacja odbierający wysłała komunikat o pozytywnym działaniu lub komunikat z negatywnym działaniem, można określić jedną lub obie z poniższych czynności.

- MQC.MQRO_PAN
- MQC.MQRO_NAN

public int TotalMessageLength {get;}

Łączna liczba bajtów w komunikacie zapisanych w kolejce komunikatów, z której ten komunikat został odebrany.

public string UserId {get; set;}

Element `UserId` jest częścią kontekstu tożsamości komunikatu. Menedżer kolejek zwykle udostępnia wartość. Wartość tę można przesłonić, jeśli użytkownik ma uprawnienia do ustawiania kontekstu tożsamości.

public int Version {get; set;}

Wersja struktury MQMD, która jest używana.

Metody komunikatów Read i Write

Metody `Read` i `Write` pełnią te same funkcje, co elementy klas `BinaryReader` i `BinaryWriter` w przestrzeni nazw `.NET System.IO`. Pełna składnia języka i przykłady użycia znajdują się w MSDN. Metody odczytują lub zapisują z bieżącej pozycji w buforze komunikatów. Przenoszą bieżącą pozycję do przodu o liczbę odczytanych lub zapisanych bajtów.

Uwaga: Jeśli dane komunikatu zawierają nagłówek `MQRFH` lub `MQRFH2`, należy użyć metody `ReadBytes`, aby odczytać dane.

- Wszystkie metody zgłaszają `IOException`.
- Metody `ReadFully` automatycznie zmień wielkość docelowej macierzy `byte` lub `sbyte` tak, aby była dokładnie zgodna z komunikatem. Zmieniana jest również tablica o wartości `NULL`.
- Metody `Read` zgłaszają `EndOfStreamException`.
- Metody `WriteDecimal` zgłaszają `MQException`.
- Metody `ReadString`, `ReadLine` i `WriteString` przekształcają się między kodami `Unicode` a zestawem znaków komunikatu. Patrz [CharacterSet](#).
- Metody `Decimal` odczytują i zapisują upakowane liczby dziesiętne kodowane w formacie `big-endian`, `MQC.MQENC_DECIMAL_NORMAL` lub `little-endian` `MQC.MQENC_DECIMAL_REVERSE`, zgodnie z wartością `Encoding`. Zakresy dziesiętne i odpowiadające im typy `.NET` są następujące:

Decimal2/short

-999 do 999

Decimal4/int

Od -99999999 do 99999999

Decimal8/long

-9999999999999999 do 9999999999999999

- Metody `Double` i `Float` odczytane i zapisują wartości zmiennopozycyjne zakodowane w formatach `big-endian` i `little endian`, `MQC.MQENC_FLOAT_IEEE_NORMAL` i `MQC.MQENC_FLOAT_IEEE_REVERSED`, lub w formacie `S/390`, `MQC.MQENC_FLOAT_S390` zgodnie z wartością `Encoding`.
- Metody `Int` odczytane i zapisują wartości całkowite zakodowane w `big-endian`, `MQC.MQENC_INTEGER_NORMAL` lub `little-endian`, `MQC.MQENC_INTEGER_REVERSED`, format, zgodnie z wartością `Encoding` (Kodowanie). Wszystkie liczby całkowite są podpisane, z wyjątkiem dodania niepodpisanego 2-bajtowego typu całkowitego. Wielkości liczb całkowitych oraz typy `.NET` i `IBM MQ` są następujące:

2 bajty

short, Int2, ushort, UInt2

4 bajt

int, Int4

8 bajtów

long, Int8

- WriteObject przenosi klasę obiektu, wartości pól nieprzejsiowych i niestatycznych oraz pola jego nadtypów, aż do buforu komunikatów.
- Program ReadObject tworzy obiekt na podstawie klasy obiektu, sygnatury klasy oraz wartości jego pól nieprzejsiowych i niestatycznych oraz pól jego nadtypów.

Tabela 219. Metody komunikatów odczytu i zapisu	
Typ docelowy	Sygnatury metod
Boolean	<pre>public bool ReadBoolean();</pre> <pre>public void WriteBoolean(bool value);</pre>
Byte	<pre>public byte ReadByte()</pre> <pre>public byte ReadUnsignedByte()</pre> <pre>public void Write(int value)</pre> <pre>public void WriteByte(int value)</pre> <pre>public void WriteByte(byte value)</pre> <pre>public void WriteByte(sbyte value)</pre>
Bytes	<pre>public byte[] ReadBytes(int count)</pre> <pre>public void ReadFully(ref byte[] value)</pre> <pre>public void ReadFully(ref sbyte[] value)</pre> <pre>public void ReadFully(ref byte[] value, int offset, int length)</pre> <pre>public void ReadFully(ref sbyte[] value, int offset, int length)</pre> <pre>public void Write(byte[] value)</pre> <pre>public void Write(sbyte[] value)</pre> <pre>public void Write(byte[] value, int offset, int length)</pre> <pre>public void Write(sbyte[] value, int offset, int length)</pre> <pre>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>

Tabela 219. Metody komunikatów odczytu i zapisu (kontynuacja)

Typ docelowy	Sygnatury metod
Double	<pre>public double ReadDouble()</pre> <pre>public void WriteDouble(double value)</pre>
Float	<pre>public float ReadFloat()</pre> <pre>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()</pre> <pre>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()</pre> <pre>public void WriteLong(long value)</pre>
Object	<pre>public Object ReadObject()</pre> <pre>public void WriteObject(Object object)</pre>
Short	<pre>public short ReadShort() public short ReadDecimal2() public short ReadInt2()</pre> <pre>public void WriteShort(int value)</pre>

Tabela 219. Metody komunikatów odczytu i zapisu (kontynuacja)

Typ docelowy	Sygnatury metod
string	<pre>public string ReadString(int length)</pre> <pre>public void WriteString(string string)</pre>
Unsigned Short	<pre>public ushort ReadUnsignedShort()</pre> <pre>public ushort ReadUInt2()</pre>
Unicode	<pre>public string ReadLine()</pre> <pre>public char ReadChar()</pre> <pre>public void WriteChar(int value)</pre> <pre>public void WriteChars(string string)</pre>
UTF	<pre>public string ReadUTF()</pre> <pre>public void WriteUTF(string string)</pre>

Metody buforowania

public void ClearMessage();

Zgłasza IOException.

Usuwa wszystkie dane w buforze komunikatów i ustawia przesunięcie danych z powrotem na zero.

public void ResizeBuffer(int size)

Zgłasza IOException.

Wskazówka do obiektu MQMessage o wielkości buforu, która może być wymagana dla kolejnych operacji pobierania. Jeśli komunikat zawiera obecnie dane komunikatu, a nowa wielkość jest mniejsza niż bieżąca wielkość, dane komunikatu są obcinane.

public void Seek(int pos)

Zgłasza IOException, ArgumentOutOfRangeException, ArgumentException.

Przesuwa kursor do pozycji bezwzględnej w buforze komunikatów podanym przez komendę *pos*. Kolejne operacje odczytu i zapisu działają na tym stanowisku w buforze.

public int SkipBytes(int i)

Zgłasza IOException, EndOfStreamException.

Przenosi do przodu n bajtów w buforze komunikatów i zwraca n, liczbę pominiętych bajtów.

Bloki metod produktu SkipBytes do momentu wystąpienia jednego z następujących zdarzeń:

- Wszystkie bajty są pomijane
- Wykryto koniec buforu komunikatów.
- Zgłoszono wyjątek

Metody właściwości

public void DeleteProperty(string name);

Zgłasza MQException.

Usuwa właściwość o określonej nazwie z komunikatu.

name

Nazwa właściwości do usunięcia.

public System.Collections.IEnumerator GetPropertyNames(string name)

Zgłasza MQException.

Zwraca IEnumerator wszystkich nazw właściwości zgodnych z podaną nazwą. Znak procentu '%' może być używany na końcu nazwy jako znak wieloznaczny w celu odfiltrowania właściwości komunikatu, dopasowywania się do zera lub większej liczby znaków, w tym okresu.

name

Nazwa właściwości, która ma zostać dopasowana.

Metody SetProperty i GetProperty

Wszystkie metody SetProperty i GetProperty zgłaszają MQException.

Metoda SetProperty klasy MQMessage .NET dodaje nową właściwość, jeśli właściwość już nie istnieje. Jeśli jednak właściwość już istnieje, podana wartość właściwości jest dodawana na końcu listy. Jeśli dla nazwy właściwości zostanie ustawiona wiele wartości przy użyciu składnika SetProperty, wywołanie funkcji GetProperty dla tej nazwy spowoduje zwrócenie tych wartości kolejno w kolejności, w jakiej zostały ustawione te wartości.

Zachowanie jest takie samo dla wszystkich metod o typie Set*Property i Get*Property, takich jak GetLongProperty, SetLongProperty, GetBooleanProperty, SetBooleanProperty, GetStringProperty i SetStringProperty.

Tabela 220. Metody SetProperty i GetProperty	
Typ	Sygnatury metod
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>

Tabela 220. Metody *SetProperty* i *GetProperty* (kontynuacja)

Typ	Sygnatury metod
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>
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>

Tabela 220. Metody *SetProperty* i *GetProperty* (kontynuacja)

Typ	Sygnatury metod
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>

Konstruktory

public MQMessage();

Tworzy obiekt MQMessage z domyślnymi informacjami o deskrytorze komunikatu i pustym buforem komunikatów.

Klasa MQProcess.NET

Za pomocą programu MQProcess można wysyłać zapytania do atrybutów procesu IBM MQ. Utwórz obiekt MQProcess przy użyciu konstruktora lub metody MQQueueManager AccessProcess.

Klasa

```
System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQProcess
```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- [“Właściwości” na stronie 1882](#)
- [“Konstruktory” na stronie 1883](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public string ApplicationId {get;}

Pobiera łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. Element ApplicationId jest używany przez aplikację monitora wyzwalacza. Element ApplicationId jest wysyłany do kolejki inicjującej jako część komunikatu wyzwalacza.

Wartością domyślną jest NULL.

public int ApplicationType {get;}

Określa typ procesu, który ma zostać uruchomiony przez aplikację monitora wyzwalacza. Typy standardowe są zdefiniowane, ale inne mogą być używane:

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

Wartością domyślną jest MQAT_NATIVE.

public string EnvironmentData {get;}

Pobiera informacje na temat środowiska aplikacji, która ma zostać uruchomiona.

Wartością domyślną jest NULL.

public string UserData {get;}

Pobiera informacje, które użytkownik udostępnił o aplikacji do uruchomienia.

Wartością domyślną jest NULL.

Konstruktory

public MQProcess(MQQueueManager queueManager, string processName, int openOptions);

public MQProcess(MQQueueManager qMgr, string processName, int openOptions, string queueManagerName, string alternateUserId);

Zgłasza MQException.

Uzyskaj dostęp do procesu IBM MQ w menedżerze kolejek *qMgr*, aby uzyskać informacje na temat atrybutów procesu.

qMgr

Menedżer kolejek do uzyskania dostępu.

processName

Nazwa procesu, który ma zostać otwarty.

openOptions

Opcje sterujące otwieraniem procesu. Poprawne opcje, które mogą zostać dodane lub połączone za pomocą bitowych OR, to:

- MQC.MQOO_FAIL_IF QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowany jest proces. Jeśli menedżer kolejek jest taki sam, jak proces, do którego uzyskiwany jest dostęp, można pozostawić pustą nazwę menedżera kolejek lub nazwę menedżera kolejek o wartości NULL.

alternateUserId

Jeśli parametr MQC.MQ00_ALTERNATE_USER_AUTHORITY jest określony w parametrze *openOptions*, *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla tego działania. Jeśli parametr MQ00_ALTERNATE_USER_AUTHORITY nie jest określony, wartość *alternateUserId* może być pusta lub mieć wartość NULL.

Domyślne uprawnienia użytkownika są używane do nawiązywania połączenia z menedżerem kolejek, jeśli nie określono programu MQC.MQ00_ALTERNATE_USER_AUTHORITY.

```
public MQProcess MQQueueManager.AccessProcess(string processName, int  
openOptions);  
public MQProcess MQQueueManager.AccessProcess(string processName, int  
openOptions, string queueManagerName, string alternateUserId);
```

Zgłasza MQException.

Uzyskaj dostęp do procesu IBM MQ w tym menedżerze kolejek, aby uzyskać informacje na temat atrybutów procesu.

processName

Nazwa procesu, który ma zostać otwarty.

openOptions

Opcje sterujące otwieraniem procesu. Poprawne opcje, które mogą zostać dodane lub połączone za pomocą bitowych OR, to:

- MQC.MQ00_FAIL_IF QUIESCING
- MQC.MQ00_INQUIRE
- MQC.MQ00_SET
- MQC.MQ00_ALTERNATE_USER_AUTHORITY

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowany jest proces. Jeśli menedżer kolejek jest taki sam, jak proces, do którego uzyskiwany jest dostęp, można pozostawić pustą nazwę menedżera kolejek lub nazwę menedżera kolejek o wartości NULL.

alternateUserId

Jeśli parametr MQC.MQ00_ALTERNATE_USER_AUTHORITY jest określony w parametrze *openOptions*, *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla tego działania. Jeśli parametr MQ00_ALTERNATE_USER_AUTHORITY nie jest określony, wartość *alternateUserId* może być pusta lub mieć wartość NULL.

Domyślne uprawnienia użytkownika są używane do nawiązywania połączenia z menedżerem kolejek, jeśli nie określono programu MQC.MQ00_ALTERNATE_USER_AUTHORITY.

Klasa MQPropertyDescriptor.NET

Parametr MQPropertyDescriptor należy używać jako parametru do metod MQMessage.GetProperty i SetProperty. MQPropertyDescriptor opisuje właściwość MQMessage.

Klasa

```
System.Object  
└─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```

- [“Właściwości” na stronie 1885](#)
- [“Konstruktor” na stronie 1886](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public int Context {get; set;}

Kontekst komunikatu, do którego należy właściwość. Dozwolone są następujące wartości:

MQC.MQPD_NO_CONTEXT

Ta właściwość nie jest powiązana z kontekstem komunikatu.

MQC.MQPD_USER_CONTEXT

Właściwość jest powiązana z kontekstem użytkownika.

Jeśli użytkownik jest autoryzowany, właściwość powiązana z kontekstem użytkownika jest zapisywana po pobraniu komunikatu. Kolejna metoda Put , która odwołuje się do zapisanego kontekstu, może przekazać tę właściwość do nowej wiadomości.

public int CopyOptions {get; set;}

CopyOptions opisuje typ komunikatu, do którego właściwość może zostać skopiowana.

Gdy menedżer kolejek odbierze komunikat zawierający zdefiniowaną właściwość IBM MQ , którą menedżer kolejek rozpoznaje jako niepoprawną, menedżer kolejek koryguje wartość pola CopyOptions .

Można określić dowolną kombinację poniższych opcji. Opcje można łączyć, dodając wartości lub używając bitowych OR.

MQC.MQCOPY_ALL

Właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

MQC.MQCOPY_FORWARD

Właściwość jest kopiowana do przekazywanego komunikatu.

MQC.MQCOPY_PUBLISH

Właściwość jest kopiowana do komunikatu odebranego przez subskrybenta, gdy jest publikowany komunikat.

MQC.MQCOPY_REPLY

Właściwość jest kopiowana do komunikatu odpowiedzi.

MQC.MQCOPY_REPORT

Właściwość jest kopiowana do komunikatu raportu.

MQC.MQCOPY_DEFAULT

Wartość nie wskazuje, że zostały określone inne opcje kopiowania. Między właściwością a kolejnymi komunikatami nie istnieje żadna relacja. Produkt MQC.MQCOPY_DEFAULT jest zawsze zwracany w przypadku właściwości deskryptora komunikatu.

MQC.MQCOPY_NONE

To samo, co MQC.MQCOPY_DEFAULT

public int Options { set; }

Opcje domyślnie: CMQC.MQPD_NONE. Nie można ustawić żadnej innej wartości.

public int Support { get; set; }

Ustaw opcję Support (Wsparcie), aby określić poziom obsługi wymagany dla właściwości komunikatu zdefiniowanych przez produkt IBM MQ. Obsługa wszystkich pozostałych właściwości jest opcjonalna. Można podać dowolną lub dowolną z następujących wartości:

MQC.MQPD_SUPPORT_OPTIONAL

Ta właściwość jest akceptowana przez menedżera kolejek nawet wtedy, gdy nie jest obsługiwana. Tę właściwość można usunąć, aby komunikat mógł przepływać do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywany do właściwości, które nie są zdefiniowane w produkcie IBM MQ .

MQC.MQPD_SUPPORT_REQUIRED

Wymagana jest obsługa właściwości. Jeśli komunikat został umieszczony w menedżerze kolejek, który nie obsługuje właściwości zdefiniowanej przez produkt IBM MQ, metoda nie powiedzie się. Zwraca kod zakończenia MQC.MQCC_FAILED i kod przyczyny MQC.MQRC_UNSUPPORTED_PROPERTY.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Obsługa właściwości jest wymagana, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Jeśli komunikat jest umieszczany w kolejce lokalnej w menedżerze kolejek, który nie obsługuje właściwości zdefiniowanej w produkcie IBM MQ, metoda nie powiedzie się. Zwraca kod zakończenia MQC.MQCC_FAILED i kod przyczyny MQC.MQRC_UNSUPPORTED_PROPERTY.

Sprawdzenie, czy komunikat jest umieszczany w zdalnym menedżerze kolejek, nie jest wykonywane.

Konstruktory

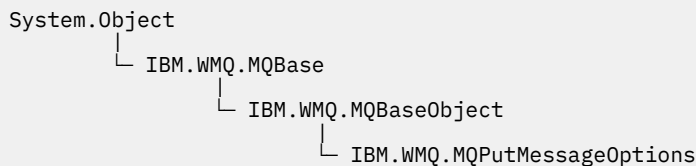
PropertyDescriptor();

Utwórz deskryptor właściwości.

Klasa MQPutMessageOptions.NET

Użyj opcji MQPutMessageOptions, aby określić sposób wysyłania komunikatów. Modyfikuje on działanie produktu MQDestination.Put.

Klasa



```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- [“Właściwości” na stronie 1886](#) [“Konstruktory” na stronie 1889](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

Uwaga: Zachowanie niektórych opcji dostępnych w tej klasie zależy od środowiska, w którym są używane. Te elementy są oznaczone gwiazdką (*).

public MQQueue ContextReference {get; set;}

Jeśli pole options zawiera MQC.MQPMO_PASS_IDENTITY_CONTEXT lub MQC.MQPMO_PASS_ALL_CONTEXT, należy ustawić to pole w taki sposób, aby odwoływało się do MQQueue, z którego mają być wyświetlane informacje o kontekście.

Wartość początkowa tego pola jest pusta.

public int InvalidDestCount {get;} *

Zazwyczaj używane dla list dystrybucyjnych, InvalidDestCount wskazuje liczbę komunikatów, których nie można było wysłać do kolejek na liście dystrybucyjnej. Liczba ta obejmuje kolejki, które nie zostały otwarte, a także kolejki, które zostały pomyślnie otwarte, ale dla których operacja put nie powiodła się.

Produkt .NET nie obsługuje list dystrybucyjnych, ale parametr InvalidDestCount jest ustawiany podczas otwierania jednej kolejki.

public int KnownDestCount {get;} *

Na ogół używane dla list dystrybucyjnych, KnownDest wskazuje liczbę komunikatów, które bieżące wywołanie pomyślnie wysłało do kolejek rozstrzyganych w kolejkach lokalnych.

Produkt .NET nie obsługuje list dystrybucyjnych, ale parametr InvalidDestCount jest ustawiany podczas otwierania jednej kolejki.

public int Options {get; set;}

Opcje sterujące działaniem produktów MQDestination.put i MQQueueManager.put. Można określić dowolną lub dowolną z poniższych wartości. Jeśli wymagana jest więcej niż jedna opcja, wartości można dodawać lub łączyć za pomocą operatora bitowego OR.

MQC.MQPMO_ASYNC_RESPONSE

Ta opcja powoduje, że wywołanie MQDestination.put jest wykonywane asynchronicznie, z niektórymi danymi odpowiedzi.

MQC.MQPMO_DEFAULT_CONTEXT

Powiąz kontekst domyślny z komunikatem.

MQC.MQPMO_FAIL_IF QUIESCING

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

MQC.MQPMO_LOGICAL_ORDER *

Umieszczanie logicznych komunikatów i segmentów w grupach komunikatów w ich kolejności logicznej.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja

MQPMO_LOGICAL_ORDER, kod przyczyny MQRC_RECONNECT_INCOMPATIBLE jest zwracany do aplikacji.

MQC.MQPMO_NEW_CORREL_ID *

Wygeneruj nowy identyfikator korelacji dla każdego wysłanego komunikatu.

MQC.MQPMO_NEW_MSG_ID *

Wygeneruj nowy identyfikator komunikatu dla każdego wysłanego komunikatu.

MQC.MQPMO_NONE

Nie określono żadnych opcji. Nie należy używać z innymi opcjami.

MQC.MQPMO_NO_CONTEXT

Z komunikatem nie ma być powiązany żaden kontekst.

MQC.MQPMO_NO_SYNCPOINT

Umieść komunikat bez elementu sterującego punktu synchronizacji. Jeśli opcja sterowania punktem synchronizacji nie jest określona, przyjmowana jest wartość domyślna bez punktu synchronizacji.

MQC.MQPMO_PASS_ALL_CONTEXT

Przekaz cały kontekst z uchwytu kolejki wejściowej.

MQC.MQPMO_PASS_IDENTITY_CONTEXT

Przekaz kontekst tożsamości z uchwytu kolejki wejściowej.

MQC.MQPMO_RESPONSE_AS_Q_DEF

W przypadku wywołania MQDestination.put ta opcja przyjmuje typ odpowiedzi typu put z atrybutu DEFPRESP kolejki.

W przypadku wywołania MQQueueManager.put ta opcja powoduje, że wywołanie jest wykonywane synchronicznie.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF

Produkt MQC.MQPMO_RESPONSE_AS_TOPIC_DEF jest synonimem produktu MQC.MQPMO_RESPONSE_AS_Q_DEF do użycia z obiektami tematów.

MQC.MQPMO_RETAIN

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Jeśli ta opcja jest używana i publikacja nie może zostać zachowana, komunikat nie zostanie opublikowany, a wywołanie zakończy się niepowodzeniem z programem MQC.MQRC_PUT_NOT_RETAINED.

Zażądaj kopii tej publikacji po jej opublikowaniu, wywołując metodę MQSubscription.RequestPublicationUpdate. Zapisana publikacja jest wysyłana do aplikacji, które tworzą subskrypcję bez ustawiania opcji MQC.MQSO_NEW_PUBLICATIONS_ONLY. Sprawdź właściwość komunikatu MQIsRetained (MQIsRetained) w publikacji, po odebraniu, aby dowiedzieć się, czy była to zachowana publikacja.

Jeśli subskrybent żąda zachowanych publikacji, użyta subskrypcja może zawierać znak wieloznaczny w łańcuchu tematu. Jeśli w drzewie tematów znajdują się wiele zachowanych publikacji, które są zgodne z subskrypcją, wszystkie te publikacje są wysyłane.

MQC.MQPMO_SET_ALL_CONTEXT

Ustaw cały kontekst z aplikacji.

MQC.MQPMO_SET_IDENTITY_CONTEXT

Ustaw kontekst tożsamości z aplikacji.

MQC.MQPMO_SYNC_RESPONSE

Ta opcja powoduje, że wywołanie MQDestination.put lub MQQueueManager.put jest wykonywane synchronicznie, z pełnymi danymi odpowiedzi.

MQC.MQPMO_SUPPRESS_REPLYTO

Wszystkie informacje wypełnione w polach ReplyToQueueName i ReplyToQueueManagerName w publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja jest używana w połączeniu z opcją raportu, która wymaga ReplyToQueueName, wywołanie kończy się niepowodzeniem z produktem MQC.MQRC_MISSING_REPLY_TO_Q.

MQC.MQPMO_SYNCPOINT

Umieść komunikat z elementem sterującym punktu synchronizacji. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

public int RecordFields {get; set;} *

Informacje o listach dystrybucyjnych. Listy dystrybucyjne nie są dostępne w produkcie .NET.

public string ResolvedQueueManagerName {get;}

Pole wyjściowe ustawione przez menedżer kolejek na nazwę menedżera kolejek, do którego należy kolejka określona przez nazwę kolejki zdalnej. ResolvedQueueManagerName może się różnić od nazwy menedżera kolejek, z którego uzyskano dostęp do kolejki, jeśli kolejka jest kolejką zdalną.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką. Jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

public string ResolvedQueueName {get;}

Pole wyjściowe ustawione przez menedżer kolejek na nazwę kolejki, w której umieszczony jest komunikat. ResolvedQueueNazwa może się różnić od nazwy użytej do otwarcia kolejki, jeśli otwarta kolejka była aliasem lub kolejką modelową.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką. Jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

public int UnknownDestCount {get;} *

Zazwyczaj używane dla list dystrybucyjnych, UnknownDestLiczebność jest polem wyjściowym ustawionym przez menedżer kolejek. Raportuje on liczbę komunikatów, które zostały pomyślnie wysłane przez bieżące wywołanie do kolejek rozstrzyganych w kolejkach zdalnych.

Produkt .NET nie obsługuje list dystrybucyjnych, ale parametr InvalidDestCount jest ustawiany podczas otwierania jednej kolejki.

Konstruktory

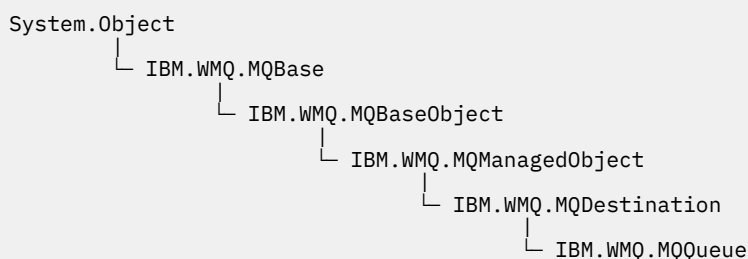
public MQPutMessageOptions();

Skonstruuj nowy obiekt MQPutMessageOptions bez ustawionego zestawu opcji, a także puste ResolvedQueueName i ResolvedQueueManagerName.

Klasa MQQueue.NET

Za pomocą programu MQQueue można wysyłać i odbierać komunikaty, a także atrybuty zapytania kolejki produktu IBM MQ. Utwórz obiekt MQQueue przy użyciu konstruktora lub metody MQQueueManager.AccessProcess.

Klasa



```
public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;
```

- [“Właściwości” na stronie 1889](#)
- [“Metody” na stronie 1891](#)
- [“Konstruktory” na stronie 1894](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public int ClusterWorkLoadPriority {get;}

Określa priorytet kolejki. Ten parametr jest poprawny tylko dla kolejek lokalnych, zdalnych i aliasowych.

public int ClusterWorkLoadRank {get;}

Określa rangę kolejki. Ten parametr jest poprawny tylko dla kolejek lokalnych, zdalnych i aliasowych.

public int ClusterWorkLoadUseQ {get;}

Określa zachowanie operacji MQPUT, gdy kolejka docelowa ma instancję lokalną i co najmniej jedną zdalną instancję klastra. Ten parametr nie ma zastosowania, jeśli operacja MQPUT pochodzi z kanału klastra. Ten parametr jest poprawny tylko dla kolejek lokalnych.

public DateTime CreationDateTime {get;}

Data i godzina utworzenia tej kolejki.

public int CurrentDepth {get;}

Pobiera liczbę komunikatów znajdujących się obecnie w kolejce. Wartość ta jest zwiększana podczas wywołania operacji put i podczas wywołania pobrania. Jest on zmniejszany podczas operacji pobierania bez przeglądania i podczas wycofywania wywołania.

public int DefinitionType {get;}

Określa, w jaki sposób kolejka została zdefiniowana. Możliwe wartości:

- MQC.MQQDT_PREDEFINED
- MQC.MQQDT_PERMANENT_DYNAMIC
- MQC.MQQDT_TEMPORARY_DYNAMIC

public int InhibitGet {get; set;}

Określa, czy możliwe jest pobieranie komunikatów w tej kolejce, czy też w tym temacie. Możliwe wartości:

- MQC.MQQA_GET_INHIBITED
- MQC.MQQA_GET_ALLOWED

public int InhibitPut {get; set;}

Określa, czy możliwe jest umieszczanie komunikatów w tej kolejce, czy też w tym temacie. Możliwe wartości:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

public int MaximumDepth {get;}

Maksymalna liczba komunikatów, które mogą znajdować się w kolejce w dowolnym momencie. Próba umieszczenia komunikatu w kolejce, która zawiera już wiele komunikatów, kończy się niepowodzeniem z kodem przyczyny MQC.MQRC_Q_FULL.

public int MaximumMessageLength {get;}

Maksymalna długość danych aplikacji, które mogą istnieć w każdym komunikacie w tej kolejce. Próba umieszczenia komunikatu większa niż ta wartość nie powiedzie się z kodem przyczyny MQC.MQRC_MSG_TOO_BIG_FOR_Q.

public int NonPersistentMessageClass {get;}

Poziom niezawodności komunikatów nietrwałych umieszczanych w tej kolejce.

public int OpenInputCount {get;}

Liczba uchwytów, które są obecnie poprawne w przypadku usuwania komunikatów z kolejki. OpenInputLiczba to łączna liczba poprawnych uchwytów danych wejściowych znanych z lokalnego menedżera kolejek, a nie tylko uchwytów utworzonych przez aplikację.

public int OpenOutputCount {get;}

Liczba uchwytów, które są obecnie poprawne w przypadku dodawania komunikatów do kolejki. OpenOutputLiczba to łączna liczba poprawnych uchwytów danych wyjściowych znanych z lokalnego menedżera kolejek, a nie tylko uchwytów utworzonych przez aplikację.

public int QueueAccounting {get;}

Określa, czy można włączyć gromadzenie informacji rozliczeniowych dla kolejki.

public int QueueMonitoring {get;}

Określa, czy możliwe jest włączenie monitorowania dla kolejki.

public int QueueStatistics {get;}

Określa, czy można włączyć gromadzenie statystyk dla kolejki.

public int QueueType {get;}

Typ tej kolejki z jedną z następujących wartości:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}

Określa, czy kolejka może być otwierana dla danych wejściowych wiele razy. Możliwe wartości:

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}

Nazwa TPIPE używana do komunikacji z OTMA przy użyciu mostu IBM MQ IMS.

public int TriggerControl {get; set;}

Określa, czy komunikaty wyzwalacza są zapisywane do kolejki inicjującej, w celu uruchomienia aplikacji w celu obsługi kolejki. Możliwe wartości:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}

Dane w formacie wolnym, które są wstawiane przez menedżera kolejek do komunikatu wyzwalacza. Wstawia on element `TriggerData`, gdy komunikat przybywający do tej kolejki powoduje zapisanie komunikatu wyzwalacza w kolejce inicjujący. Maksymalna dopuszczalna długość łańcucha jest podawana przez `MQC.MQ_TRIGGER_DATA_LENGTH`.

public int TriggerDepth {get; set;}

Liczba komunikatów, które muszą znajdować się w kolejce, zanim zostanie zapisany komunikat wyzwalacza, gdy typ wyzwalacza jest ustawiony na wartość `MQC.MQTT_DEPTH`.

public int TriggerMessagePriority {get; set;}

Priorytet komunikatu, pod którym komunikaty nie mają udziału w generowaniu komunikatów wyzwalacza. Oznacza to, że menedżer kolejek ignoruje te komunikaty podczas podejmowania decyzji o tym, czy ma zostać wygenerowany wyzwalacz. Wartość zero powoduje, że wszystkie komunikaty mogą przyczyniać się do generowania komunikatów wyzwalacza.

public int TriggerType {get; set;}

Warunki, w których komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki. Możliwe wartości:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT EVERY
- MQC.MQTT_DEPTH

Metody

public void Get(MQMessage message);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);

public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int MaxMsgSize);

Zgłasza `MQException`.

Pobiera komunikat z kolejki.

Jeśli operacja pobierania nie powiedzie się, obiekt `MQMessage` nie zostanie zmieniony. Jeśli operacja powiedzie się, deskryptor komunikatu i fragmenty danych komunikatu produktu `MQMessage` zostaną zastąpione przez deskryptor komunikatu i dane komunikatu z komunikatu przychodzącego.

Wszystkie wywołania programu IBM MQ z określonego `MQQueueManager` są synchroniczne. Z tego powodu, jeśli zostanie wykonane oczekiwanie, wszystkie inne wątki korzystające z tego samego serwera `MQQueueManager` są blokowane z kolejnych wywołań programu IBM MQ do czasu zakończenia operacji `Get`. Jeśli wymagane jest jednoczesne uzyskiwanie dostępu do produktu IBM MQ z wielu wątków, każdy wątek musi utworzyć własny obiekt `MQQueueManager`.

message

Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskrypcorze komunikatu są parametrami wejściowymi. Ważne jest, aby parametry wejściowe `MessageId` i `CorrelationId` zostały ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny `MQRC_BACKED_OUT` po pomyślnym ponownym nawiązaniu połączenia, dla komunikatów odebranych w ramach produktu `MQGM_SYNCPOINT`.

getMessageOptions

Opcje sterujące działaniem `get`.

Użycie opcji `MQC.MQGMO_CONVERT` może spowodować wystąpienie wyjątku z kodem przyczyny `MQC.MQRC_CONVERTED_STRING_TOO_BIG` podczas przekształcania z jednobajtowych kodów

znaków na kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie zostanie określony, użyta zostanie opcja MQGMO_NOWAIT.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQGMO_LOGICAL_ORDER, zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE.

MaxMsgSize

Największa wiadomość, którą ten obiekt komunikatu ma odebrać. Jeśli komunikat w kolejce jest większy niż ten rozmiar, występuje jedna z dwóch rzeczy:

- Jeśli flaga MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions, to komunikat jest wypełniany możliwie jak największą ilością danych komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_ACCEPTED.
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_FAILED.

Jeśli parametr *MaxMsgSize* nie zostanie określony, zostanie pobrany cały komunikat.

public void Put(MQMessage message);
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

Zgłasza MQException.

Umieszcza komunikat w kolejce.

Modyfikacje obiektu MQMessage po zakończeniu wywołania Put nie mają wpływu na rzeczywisty komunikat w kolejce IBM MQ ani w temacie publikacji.

Produkt Put aktualizuje właściwości MessageId i CorrelationId obiektu MQMessage i nie powoduje czyszczenia danych komunikatu. Dalsze wywołania programu Put lub Get odnoszą się do zaktualizowanych informacji w obiekcie MQMessage. Na przykład w następującym fragmencie kodu pierwszy komunikat zawiera a i drugi ab.

```
msg.WriteString("a");  
q.Put(msg, pmo);  
msg.WriteString("b");  
q.Put(msg, pmo);
```

message

Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat, który ma zostać wysłany. W wyniku tej metody deskryptor komunikatu może zostać zmieniony. Wartości w deskrytorze komunikatu natychmiast po zakończeniu tej metody to wartości, które zostały umieszczone w kolejce lub opublikowane w temacie.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- MQRC_CALL_INTERRUPTED, jeśli połączenie zostało zerwane podczas wykonywania wywołania Put w trwałym komunikacie, a ponowne nawiązanie połączenia powiodło się.
- MQRC_NONE, jeśli połączenie jest pomyślne podczas uruchamiania wywołania Put w nietrwałym komunikacie (patrz sekcja [Odtwarzanie aplikacji](#)).

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: Aby uprościć i uzyskać wydajność, należy użyć obiektu MQQueueManager.Put , aby umieścić pojedynczy komunikat w kolejce. W tym celu należy mieć dla niego obiekt MQQueue .

```
public void PutForwardMessage(MQMessage message);  
public void PutForwardMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions);
```

Zgłasza MQException

Umieść komunikat przesyłany do kolejki, gdzie *message* jest pierwotnym komunikatem.

message

Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat, który ma zostać wysłany. W wyniku tej metody deskryptor komunikatu może zostać zmieniony. Wartości w deskrytorze komunikatu natychmiast po zakończeniu tej metody to wartości, które zostały umieszczone w kolejce lub opublikowane w temacie.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania Put w trwałym komunikacie, a ponowne nawiązanie połączenia powiodło się.
- MQRC_NONE , jeśli połączenie jest pomyślne podczas uruchamiania wywołania Put w nietrwałym komunikacie (patrz sekcja [Odtwarzanie aplikacji](#)).

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions .

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

```
public void PutReplyMessage(MQMessage message)  
public void PutReplyMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

Zgłasza MQException.

Umieść komunikat odpowiedzi w kolejce, gdzie *message* jest pierwotnym komunikatem.

message

Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskrytorze komunikatu są parametrami wejściowymi. Ważne jest, aby parametry wejściowe MessageId i CorrelationId zostały ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia, dla komunikatów odebranych w ramach produktu MQGM_SYNCPOINT.

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions .

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

```
public void PutReportMessage(MQMessage message)  
public void PutReportMessage(MQMessage message, MQPutMessageOptions  
putMessageOptions)
```

Zgłasza MQException.

Umieść komunikat raportu w kolejce, gdzie *message* jest pierwotnym komunikatem.

message

Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskrytorze komunikatu są parametrami wejściowymi. Ważne jest, aby parametry wejściowe MessageId i CorrelationId zostały ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia, dla komunikatów odebranych w ramach produktu MQGM_SYNCPOINT.

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions .

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

Konstruktory

```
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);  
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

Zgłasza MQException.

Uzyskuje dostęp do kolejki w tym menedżerze kolejek.

Użytkownik może uzyskać lub przeglądać komunikaty, umieszczać komunikaty, pytać o atrybuty kolejki lub ustawiać atrybuty kolejki. Jeśli kolejka nazwana jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wyślij zapytanie do atrybutu name wynikowego obiektu MQQueue , aby dowiedzieć się, jak nazwa kolejki dynamicznej jest określona.

queueName

Nazwa kolejki do otwarcia.

openOptions

Opcje sterujące otwieraniem kolejki.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

MQC.MQOO_BIND_AS_QDEF

Użyj domyślnego powiązania dla kolejki.

MQC.MQOO_BIND_NOT_FIXED

Nie należy wiązać się z konkretnym miejscem docelowym.

MQC.MQOO_BIND_ON_OPEN

Powiąz uchwyty z miejscem docelowym, gdy kolejka jest otwierana.

MQC.MQOO_BROWSE

Otwórz, aby przeglądać wiadomość.

MQC.MQOO_FAIL_IF QUIESCING

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

MQC.MQOO_INPUT_AS_Q_DEF

Otwórz, aby uzyskać komunikaty przy użyciu wartości domyślnej zdefiniowanej przez kolejkę.

MQC.MQOO_INPUT_SHARED

Otwórz, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

MQC.MQOO_INPUT_EXCLUSIVE

Otwórz, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

MQC.MQOO_INQUIRE

Otwórz do zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT

Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT

Zezwól na przekazanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Zezwalaj na przekazanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT

Zapisz kontekst podczas pobierania komunikatu.

MQC.MQOO_SET

Otwórz, aby ustawić atrybuty-wymagane, jeśli mają zostać ustawione właściwości.

MQC.MQOO_SET_ALL_CONTEXT

Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT

Umożliwia ustawienie kontekstu tożsamości.

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowana jest kolejka. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżer kolejek, z którym połączony jest obiekt `MQQueueManager`.

dynamicQueueName

Parametr *dynamicQueueName* jest ignorowany, chyba że parametr *queueName* określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli *queueName* określa nazwę kolejki modelowej. Jeśli ostatni niepusty znak w nazwie to gwiazdka (*), to menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w tym menedżerze kolejek.

alternateUserId

Jeśli parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` jest określony w parametrze `openOptions`, to *alternateUserId* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla otwarcia. Jeśli parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nie jest określony, wartość *alternateUserId* może pozostać pusta lub mieć wartość NULL.

```
public MQQueue(MQQueueManager queueManager, string queueName, int openOptions,  
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

Zgłasza `MQException`.

Uzyskuje dostęp do kolejki w systemie `queueManager`.

Użytkownik może uzyskać lub przeglądać komunikaty, umieszczać komunikaty, pytać o atrybuty kolejki lub ustawiać atrybuty kolejki. Jeśli kolejka nazwana jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wyślij zapytanie do atrybutu `name` wynikowego obiektu `MQQueue`, aby dowiedzieć się, jak nazwa kolejki dynamicznej jest określona.

queueManager

Menedżer kolejek w celu uzyskania dostępu do kolejki.

queueName

Nazwa kolejki do otwarcia.

openOptions

Opcje sterujące otwieraniem kolejki.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

MQC.MQOO_BIND_AS_QDEF

Użyj domyślnego powiązania dla kolejki.

MQC.MQOO_BIND_NOT_FIXED

Nie należy wiązać się z konkretnym miejscem docelowym.

MQC.MQOO_BIND_ON_OPEN

Powiązaj uchwyt z miejscem docelowym, gdy kolejka jest otwierana.

MQC.MQOO_BROWSE

Otwórz, aby przeglądać wiadomość.

MQC.MQOO_FAIL_IF QUIESCING

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

MQC.MQOO_INPUT_AS_Q_DEF

Otwórz, aby uzyskać komunikaty przy użyciu wartości domyślnej zdefiniowanej przez kolejkę.

MQC.MQOO_INPUT_SHARED

Otwórz, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

MQC.MQOO_INPUT_EXCLUSIVE

Otwórz, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

MQC.MQOO_INQUIRE

Otwórz do zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT

Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT

Zezwól na przekazanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Zezwalaj na przekazanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT

Zapisz kontekst podczas pobierania komunikatu.

MQC.MQOO_SET

Otwórz, aby ustawić atrybuty-wymagane, jeśli mają zostać ustawione właściwości.

MQC.MQOO_SET_ALL_CONTEXT

Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT

Umożliwia ustawienie kontekstu tożsamości.

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowana jest kolejka. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżer kolejek, z którym połączony jest obiekt MQQueueManager .

dynamicQueueName

Parametr *dynamicQueueName* jest ignorowany, chyba że parametr *queueName* określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli *queueName* określa nazwę kolejki modelowej. Jeśli ostatni niepusty znak w nazwie to gwiazdka (*), to menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w tym menedżerze kolejek.

alternateUserId

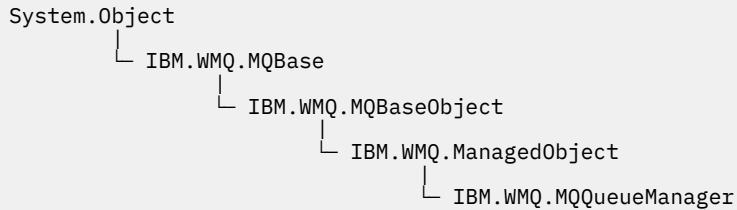
Jeśli parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY jest określony w parametrze *openOptions* , to *alternateUserId* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla otwarcia. Jeśli parametr

MQC.MQ00_ALTERNATE_USER_AUTHORITY nie jest określony, wartość *alternateUserId* może pozostać pusta lub mieć wartość NULL.

Klasa MQQueueManager.NET

Program MQQueueManager służy do nawiązywania połączeń z menedżerem kolejek i obiektami menedżera kolejek. Kontroluje również transakcje. Konstruktor MQQueueManager tworzy połączenie z klientem lub serwerem.

Klasa



```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```

- [“Właściwości” na stronie 1897](#)
- [“Metody” na stronie 1900](#)
- [“Konstruktory” na stronie 1906](#)

Właściwości

Test dla MQException zgłaszanego podczas pobierania właściwości.

public int AccountingConnOverride {get;}

Określa, czy aplikacje mogą przestonić ustawienie wartości rozliczania i rozliczania kolejki MQI .

public int AccountingInterval {get;}

Jak długo przed zapisami pośrednich zapisów księgowych (w sekundach).

public int ActivityRecording {get;}

Steruje generowaniem raportów działania.

public int AdoptNewMCACheck {get;}

Określa, które elementy są sprawdzane w celu określenia, czy agent MCA ma zostać adoptowane po wykryciu nowego kanału danych przychodzących. Aby nazwa agenta MCA została przyjęta, musi być zgodna z nazwą aktywnego agenta MCA.

public int AdoptNewMCAInterval {get;}

Czas (w sekundach), przez jaki nowy kanał oczekuje na zakończenie osieroconego kanału.

public int AdoptNewMCAType {get;}

Określa, czy osierocona instancja MCA ma zostać adoptowana (zrestartowana), gdy wykryto nowe żądanie kanału danych przychodzących zgodne z wartością MCACheck AdoptNew.

public int BridgeEvent {get;}

Określa, czy generowane są zdarzenia mostu IMS .

public int ChannelEvent {get;}

Określa, czy generowane są zdarzenia kanału.

public int ChannelInitiatorControl {get;}

Określa, czy inicjator kanału jest uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

public int ChannelInitiatorAdapters {get;}

Liczba podzadań adaptera do przetwarzania wywołań IBM MQ .

public int ChannelInitiatorDispatchers {get;}

Liczba programów rozsyłających, które mają zostać użyte dla inicjatora kanału.

public int ChannelInitiatorTraceAutoStart {get;}

Określa, czy śledzenie inicjatora kanału jest uruchamiane automatycznie.

public int ChannelInitiatorTraceTableSize {get;}

Wielkość (w megabajtach) obszaru danych śledzenia inicjatora kanału.

public int ChannelMonitoring {get;}

Określa, czy jest używany monitorowanie kanałów.

public int ChannelStatistics {get;}

Steruje kolekcjonowaniem danych statystycznych dla kanałów.

public int CharacterSet {get;}

Zwraca identyfikator kodowanego zestawu znaków (CCSID) menedżera kolejek. Element CharSet jest używany przez menedżer kolejek dla wszystkich pól łańcucha znaków w interfejsie programistycznym aplikacji.

public int ClusterSenderMonitoring {get;}

Steruje gromadzeniem danych monitorowania w trybie z połączeniem dla automatycznie zdefiniowanych kanałów nadajnika klastrów.

public int ClusterSenderStatistics {get;}

Steruje gromadzeniem danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów.

public int ClusterWorkLoadMRU {get;}

Maksymalna liczba wychodzących kanałów klastra.

public int ClusterWorkLoadUseQ {get;}

Wartość domyślna właściwości MQQueue , ClusterWorkLoadUseQ, jeśli określa ona wartość QMGR.

public int CommandEvent {get;}

Określa, czy generowane są zdarzenia komend.

public string CommandInputQueueName {get;}

Zwraca nazwę kolejki wejściowej komend zdefiniowanej w menedżerze kolejek. Aplikacje mogą wysyłać komendy do tej kolejki, jeśli jest to autoryzowane.

public int CommandLevel {get;}

Wskazuje poziom funkcji menedżera kolejek. Zestaw funkcji, które odpowiadają poszczególnym poziomom funkcji, zależy od platformy. Na konkretnej platformie można polegać na każdym menedżerze kolejek, który obsługuje funkcje na najniższym poziomie funkcyjnym, które są wspólne dla wszystkich menedżerów kolejek.

public int CommandLevel {get;}

Określa, czy serwer komend jest uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

public string DNSGroup {get;}

Nie jest już używany.

public int DNSWLM {get;}

Nie jest już używany.

public int IPAddressVersion {get;}

Który protokół IP (IPv4 lub IPv6) ma być używany dla połączenia kanału.

public boolean IsConnected {get;}

Zwraca wartość parametru isConnected.

Wartość true oznacza, że połączenie z menedżerem kolejek zostało nawiązane i nie jest znane jako zerwane. Wszystkie wywołania funkcji IsConnected nie podejmują aktywnego działania w celu uzyskania dostępu do menedżera kolejek, dlatego możliwe jest przerwanie połączenia fizycznego, ale IsConnected nadal może zwrócić wartość true. Stan IsConnected (Połączono) jest aktualizowany tylko wtedy, gdy działanie, na przykład umieszczanie komunikatu, uzyskiwanie komunikatu, jest wykonywane w menedżerze kolejek.

Wartość false oznacza, że połączenie z menedżerem kolejek nie zostało nawiązane lub zostało zerwane lub zostało rozłączone.

public int KeepAlive {get;}

Określa, czy narzędzie TCP KEEPALIVE ma być używane do sprawdzania, czy drugi koniec połączenia jest nadal dostępny. Jeśli jest on niedostępny, kanał jest zamknięty.

public int ListenerTimer {get;}

Odstęp czasu (w sekundach) między kolejnymi próbami zrestartowania obiektu nasłuchiwanego przez program IBM MQ po awarii APPC lub TCP/IP.

public int LoggerEvent {get;}

Określa, czy generowane są zdarzenia programu rejestrującego.

public string LU62ARMSuffix {get;}

Przyrostek elementu APPCPM systemu SYS1.PARMLIB. Przyrostek wyznacza LUADD do inicjatora kanału. Gdy menedżer automatycznego restartu (ARM) restartuje inicjator kanału, uruchamiana jest komenda z/OS SET APPC=xx.

public string LUGroupName {get; z/os}

Ogólna nazwa LU, która ma być używana przez program nasłuchujący LU 6.2 obsługujący transmisje przychodzące dla grupy współużytkowania kolejki.

public string LUName {get;}

Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 .

public int MaximumActiveChannels {get;}

Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie.

public int MaximumCurrentChannels {get;}

Maksymalna liczba kanałów, które mogą być aktualne w dowolnym momencie (w tym kanały połączenia z serwerem z połączonymi klientami).

public int MaximumLU62Channels {get;}

Maksymalna liczba kanałów, które mogą być bieżące lub klienci, które mogą być podłączone, które korzystają z protokołu transmisji LU 6.2 .

public int MaximumMessageLength {get;}

Zwraca maksymalną długość komunikatu (w bajtach), który może być obsługiwany przez menedżer kolejek. Kolejka nie może być zdefiniowana z maksymalną długością komunikatu większą niż MaximumMessageLength.

public int MaximumPriority {get;}

Zwraca maksymalny priorytet komunikatu obsługiwany przez menedżer kolejek. Priorytety zakresu od zera (najniższy) do tej wartości. Zgłasza MQException po wywołaniu tej metody po odłączeniu od menedżera kolejek.

public int MaximumTCPChannels {get;}

Maksymalna liczba kanałów, które mogą być bieżące, lub klientów, które mogą być podłączone, które korzystają z protokołu transmisji TCP/IP.

public int MQIAccounting {get;}

Steruje kolekcjonowaniem informacji rozliczeniowych dla danych MQI.

public int MQIStatistics {get;}

Steruje kolekcjonowaniem informacji monitorowania statystyk dla menedżera kolejek.

public int OutboundPortMax {get;}

Maksymalna wartość z zakresu numerów portów, która ma być używana podczas wiązania kanałów wychodzących.

public int OutboundPortMin {get;}

Minimalna wartość z zakresu numerów portów, która ma być używana podczas wiązania kanałów wychodzących.

public int QueueAccounting {get;}

Określa, czy dane rozliczeniowe klasy 3 (rozliczanie na poziomie wątku i na poziomie kolejki) mają być używane dla wszystkich kolejek.

public int QueueMonitoring {get;}

Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek.

public int QueueStatistics {get;}

Steruje kolekcjonowaniem danych statystycznych dla kolejek.

public int ReceiveTimeout {get;}

Czas, przez jaki kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od swojego partnera przed powrotem do stanu nieaktywnego.

public int ReceiveTimeoutMin {get;}

Minimalny czas, przez jaki kanał TCP/IP oczekuje na odbiór danych, w tym pulsy, od swojego partnera przed powrotem do stanu nieaktywnego.

public int ReceiveTimeoutType {get;}

Kwalifikator, który ma zostać zastosowany do wartości w pliku ReceiveTimeout.

public int SharedQueueQueueManagerName {get;}

Określa sposób dostarczania komunikatów do kolejki współużytkowanej. Jeśli właściwość put określa inny menedżer kolejek z tej samej grupy współużytkowania kolejki, co docelowy menedżer kolejek, komunikat jest dostarczany na dwa sposoby:

MQC.MQSQQM_USE

Komunikaty są dostarczane do menedżera kolejek obiektów przed umieszczenie ich w kolejce współużytkowanej.

MQCMQSQQM_IGNORE

Komunikaty są umieszczane bezpośrednio w kolejce współużytkowanej.

public int SSLEvent {get;}

Określa, czy zdarzenia SSL są generowane.

public int SSLFips {get;}

Określa, czy tylko algorytmy certyfikowane przez FIPS mają być używane, jeśli kryptografia jest wykonywana w produkcie IBM MQ, a nie w sprzęcie kryptograficznym.

public int SSLKeyResetCount {get;}

Wskazuje liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed renegecją klucza tajnego.

public int ClusterSenderStatistics {get;}

Określa odstęp czasu (w minutach) między kolejnymi zbieraniem statystyk.

public int SyncpointAvailability {get;}

Wskazuje, czy menedżer kolejek obsługuje jednostki pracy i punkty synchronizacji przy użyciu metod MQQueue.get i MQQueue.put.

public string TCPName {get;}

Nazwa jedynego lub domyślnego systemu TCP/IP, który ma być używany, w zależności od wartości parametru TCPStackType.

public int TCPStackType {get;}

Określa, czy inicjator kanału używa tylko przestrzeni adresowej TCP/IP określonej w opcji TCPName. Alternatywnie, inicjator kanału może wiązać się z dowolnym adresem TCP/IP.

public int TraceRouteRecording {get;}

Steruje rejestrowaniem informacji o śledzeniu trasy.

Metody

public MQProcess AccessProcess(string processName, int openOptions);

public MQProcess AccessProcess(string processName, int openOptions, string queueManagerName, string alternateUserId);

Zgłasza MQException.

Uzyskaj dostęp do procesu IBM MQ w tym menedżerze kolejek, aby uzyskać informacje na temat atrybutów procesu.

processName

Nazwa procesu, który ma zostać otwarty.

openOptions

Opcje sterujące otwieraniem procesu. Poprawne opcje, które mogą zostać dodane lub połączone za pomocą bitowych OR, to:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowany jest proces. Jeśli menedżer kolejek jest taki sam, jak proces, do którego uzyskiwany jest dostęp, można pozostawić pustą nazwę menedżera kolejek lub nazwę menedżera kolejek o wartości NULL.

alternateUserId

Jeśli parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY jest określony w parametrze *openOptions*, *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla tego działania. Jeśli parametr MQOO_ALTERNATE_USER_AUTHORITY nie jest określony, wartość *alternateUserId* może być pusta lub mieć wartość NULL.

Domyślne uprawnienia użytkownika są używane do nawiązywania połączenia z menedżerem kolejek, jeśli nie określono programu MQC.MQOO_ALTERNATE_USER_AUTHORITY.

```
public MQQueue AccessQueue(string queueName, int openOptions);  
public MQQueue AccessQueue(string queueName, int openOptions, string  
queueManagerName, string dynamicQueueName, string alternateUserId);
```

Zgłasza MQException.

Uzyskuje dostęp do kolejki w tym menedżerze kolejek.

Użytkownik może uzyskać lub przeglądać komunikaty, umieszczać komunikaty, pytać o atrybuty kolejki lub ustawiać atrybuty kolejki. Jeśli kolejka nazwana jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wyślij zapytanie do atrybutu name wynikowego obiektu MQQueue, aby dowiedzieć się, jak nazwa kolejki dynamicznej jest określona.

queueName

Nazwa kolejki do otwarcia.

openOptions

Opcje sterujące otwieraniem kolejki.

MQC.MQOO_ALTERNATE_USER_AUTHORITY

Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

MQC.MQOO_BIND_AS_QDEF

Użyj domyślnego powiązania dla kolejki.

MQC.MQOO_BIND_NOT_FIXED

Nie należy wiązać się z konkretnym miejscem docelowym.

MQC.MQOO_BIND_ON_OPEN

Powiąż uchwyt z miejscem docelowym, gdy kolejka jest otwierana.

MQC.MQOO_BROWSE

Otwórz, aby przeglądać wiadomość.

MQC.MQOO_FAIL_IF_QUIESCING

Niepowodzenie, jeśli menedżer kolejek jest wygaszany.

MQC.MQOO_INPUT_AS_Q_DEF

Otwórz, aby uzyskać komunikaty przy użyciu wartości domyślnej zdefiniowanej przez kolejkę.

MQC.MQOO_INPUT_SHARED

Otwórz, aby uzyskać dostęp do komunikatów z dostępem współużytkowanym.

MQC.MQOO_INPUT_EXCLUSIVE

Otwórz, aby uzyskać dostęp do komunikatów z wyłącznym dostępem.

MQC.MQOO_INQUIRE

Otwórz do zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT

Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT

Zezwól na przekazanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT

Zezwalaj na przekazanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT

Zapisz kontekst podczas pobierania komunikatu.

MQC.MQOO_SET

Otwórz, aby ustawić atrybuty-wymagane, jeśli mają zostać ustawione właściwości.

MQC.MQOO_SET_ALL_CONTEXT

Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT

Umożliwia ustawienie kontekstu tożsamości.

queueManagerName

Nazwa menedżera kolejek, w którym zdefiniowana jest kolejka. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżer kolejek, z którym połączony jest obiekt MQQueueManager.

dynamicQueueName

Parametr *dynamicQueueName* jest ignorowany, chyba że parametr *queueName* określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli *queueName* określa nazwę kolejki modelowej. Jeśli ostatni niepusty znak w nazwie to gwiazdka (*), to menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w tym menedżerze kolejek.

alternateUserId

Jeśli parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY jest określony w parametrze *openOptions*, to *alternateUserId* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla otwarcia. Jeśli parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY nie jest określony, wartość *alternateUserId* może pozostać pusta lub mieć wartość NULL.

```

public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options, string alternateUserId);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName, System.Collections.Hashtable
properties);

```

Uzyskaj dostęp do tematu dotyczącego tego menedżera kolejek.

Obiekty produktu MQTopic są ściśle powiązane z obiektami tematu administracyjnego, które są czasami nazywane obiektami tematów. Na wejściu element topicObject wskazuje na obiekt tematu administracyjnego. Konstruktor MQTopic uzyskuje łańcuch tematu z obiektu tematu i łączy go z topicName w celu utworzenia nazwy tematu. Albo oba obiekty topicObject lub topicName mogą mieć wartość NULL. Nazwa tematu zostanie dopasowana do drzewa tematów, a nazwa najbliższego zgodnego obiektu tematu administracyjnego jest zwracana w obiekcie topicObject.

Tematy powiązane z obiektem MQTopic są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez produkt *topicObject*. Drugi łańcuch tematu to *topicString*. Wynikowy łańcuch tematu powiązany z obiektem MQTopic może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwierany na potrzeby publikowania lub subskrybowania, można użyć metod MQTopic .Put do publikowania w tematach lub metod MQTopic .Get w celu otrzymywania publikacji na tematy. Aby opublikować ten sam temat i zasubskrybować ten sam temat, należy dwukrotnie uzyskać dostęp do tematu, raz na potrzeby publikowania i subskrypcji.

W przypadku utworzenia obiektu MQTopic dla subskrypcji, bez udostępniania obiektu MQDestination , zakłada się subskrypcję zarządzaną. Jeśli kolejka jest zaliczana jako obiekt MQDestination , zakłada się, że subskrypcja niezarządzana jest niezarządzana. Należy upewnić się, że ustawione opcje subskrypcji są spójne z subskrypcją zarządzaną lub niezarządzaną.

destination

destination jest instancją produktu MQQueue . Udostępniając produkt *destination*, produkt MQTopic jest otwierany jako subskrypcja niezarządzana. Publikacje dotyczące tego tematu są dostarczane do kolejki, do której dostęp jest uzyskiwany jako *destination*.

topicName

Łańcuch tematu, który jest drugą częścią nazwy tematu. *topicName* jest konkatelowany z łańcuchem tematu zdefiniowanym w administracyjnym obiekcie tematu *topicObject* . Parametr *topicName* można ustawić na wartość NULL, w którym to przypadku nazwa tematu jest definiowana przez łańcuch tematu w produkcie *topicObject*.

topicObject

Na wejściu *topicObject* jest nazwą obiektu tematu, który zawiera łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w produkcie *topicObject* jest konkatelowany z produktem *topicName*. Reguły konstruowania łańcuchów tematów są zdefiniowane w sekcji [łączenie łańcuchów tematów](#).

Na wyjściu program *topicObject* zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zbliżony w drzewie tematów do tematu określonego przez łańcuch tematu.

openAs

Przejdź do tematu, aby opublikować lub zasubskrybować. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

W tym celu należy połączyć opcje sterujące otwieraniem tematu w celu ich publikacji lub subskrypcji. Użyj stałych MQC.MQSO_*, aby uzyskać dostęp do tematu dotyczącego stałych subskrypcji i MQC.MQOO_*, aby uzyskać dostęp do tematu w celu opublikowania.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości razem lub połącz wartości opcji przy użyciu operatora OR bitowego.

alternateUserId

Podaj alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji wymaganej do zakończenia operacji. Wartość *alternateUserId* należy określić, jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY.

subscriptionName

subscriptionName jest wymagany, jeśli dostępne są opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER. W obu przypadkach produkt MQTopic jest niejawnie otwarty dla subskrypcji. Wyjątek jest zgłaszany, jeśli ustawiono MQC.MQSO_DURABLE, a subskrypcja istnieje, lub jeśli ustawiona jest wartość MQC.MQSO_ALTER, a subskrypcja nie istnieje.

properties

Ustaw dowolną z wymienionych właściwości subskrypcji przy użyciu tabeli mieszającej. Określone pozycje w tabeli mieszającej są aktualizowane z wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQAsyncStatus GetAsyncStatus();

Zgłasza MQException

Zwraca obiekt MQAsyncStatus, który reprezentuje działanie asynchroniczne dla połączenia menedżera kolejek.

public void Backout();

Zgłasza MQException.

Wycofuje wszystkie komunikaty, które zostały odczytane lub zapisane w punkcie synchronizacji od ostatniego punktu synchronizacji.

Komunikaty, które zostały zapisane z ustawioną flagą MQC.MQPMO_SYNCPOINT, są usuwane z kolejek. Komunikaty odczytywane z opcją MQC.MQGMO_SYNCPOINT są przywrócone do kolejek, z których pochodzą. Jeśli komunikaty są trwałe, zmiany są rejestrowane.

W przypadku klientów z możliwością ponownego połączenia kod przyczyny produktu MQRC_NONE jest zwracany do klienta po pomyślnym nawiązaniu połączenia.

public void Begin();

Zgłasza MQException.

Produkt Begin jest obsługiwany tylko w trybie powiązań serwera. Uruchamia globalną jednostkę pracy.

public void Commit();

Zgłasza MQException.

Zatwierdź wszystkie komunikaty, które zostały odczytane lub zapisane w punkcie synchronizacji od ostatniego punktu synchronizacji.

Komunikaty zapisane z ustawioną flagą MQC.MQPMO_SYNCPOINT są dostępne dla innych aplikacji. Komunikaty pobrane z ustawioną flagą MQC.MQGMO_SYNCPOINT są usuwane. Jeśli komunikaty są trwałe, zmiany są rejestrowane.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało utracone podczas wykonywania wywołania zatwierdzenia.
- MQRC_BACKED_OUT , jeśli wywołanie zatwierdzenia zostało wydane po ponownym nawiązaniu połączenia.

Disconnect();

Zgłasza MQException.

Zamknij połączenie z menedżerem kolejek. Wszystkie obiekty, do których uzyskano dostęp w tym menedżerze kolejek, nie są już dostępne dla tej aplikacji. Aby ponownie uzyskać dostęp do obiektów, należy utworzyć obiekt MQQueueManager .

Ogólnie, każda praca wykonana jako część jednostki pracy jest zatwierdzana. Jeśli jednak jednostka pracy jest zarządzana przez produkt .NET, jednostka pracy może zostać wycofana.

```
public void Put(int type, string destinationName, MQMessage message);  
public void Put(int type, string destinationName, MQMessage message  
MQPutMessageOptions putMessageOptions);  
public void Put(int type, string destinationName, string queueManagerName,  
string topicString, MQMessage message);  
public void Put(string queueName, MQMessage message);  
public void Put(string queueName, MQMessage message, MQPutMessageOptions  
putMessageOptions);  
public void Put(string queueName, string queueManagerName, MQMessage message);  
public void Put(string queueName, string queueManagerName, MQMessage message,  
MQPutMessageOptions putMessageOptions);  
public void Put(string queueName, string queueManagerName, MQMessage message,  
MQPutMessageOptions putMessageOptions, string alternateUserId);
```

Zgłasza MQException.

Umieszcza pojedynczy komunikat w kolejce lub temacie bez tworzenia obiektu MQQueue lub MQTopic jako pierwszy.

queueName

Nazwa kolejki, na której ma zostać umieszczony komunikat.

destinationName

Nazwa obiektu docelowego. Jest to kolejka lub temat w zależności od wartości *type*.

type

Typ obiektu docelowego. Nie można łączyć opcji.

MQC.MQOT_Q

Kolejka

MQC.MQOT_TOPIC

Temat

queueManagerName

Nazwa menedżera kolejek lub aliasu menedżera kolejek, w którym zdefiniowana jest kolejka. Jeśli określono typ MQC.MQOT_TOPIC, ten parametr jest ignorowany.

Jeśli kolejka jest kolejką modelową, a rozstrzygniętą nazwą menedżera kolejek nie jest ten menedżer kolejek, zgłaszany jest MQException.

topicString

Produkt *topicString* jest łączony z nazwą tematu w obiekcie tematu *destinationName*.

topicString jest ignorowany, jeśli *destinationName* jest kolejką.

message

Komunikat do wysłania. Komunikat jest obiektem wejścia/wyjścia.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- MQRC_CALL_INTERRUPTED, jeśli połączenie zostało zerwane podczas wykonywania wywołania Put w przypadku komunikatu trwałego.
- MQRC_NONE, jeśli połączenie jest pomyślne podczas wykonywania wywołania Put w nietrwałym komunikacie (patrz sekcja [Odtwarzanie aplikacji](#)).

putMessageOptions

Opcje sterujące działaniami operacji put.

Jeśli parametr *putMessageOptions* zostanie pominięty, zostanie utworzona domyślna instancja produktu *putMessageOptions*. *putMessageOptions* jest obiektem wejścia/wyjścia.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER, zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE.

alternateUserId

Określa alternatywny identyfikator użytkownika używany do sprawdzania autoryzacji podczas umieszczania komunikatu w kolejce.

Opcję *alternateUserId* można pominąć, jeśli w produkcie *putMessageOptions* nie jest ustawiona wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY. Jeśli zostanie ustawiona wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY, należy również ustawić wartość *alternateUserId*. *alternateUserId* nie działa, chyba że zostanie również ustawiony parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY.

Konstruktory

```
public MQQueueManager();  
public MQQueueManager(string queueManagerName);  
public MQQueueManager(string queueManagerName, Int options);  
public MQQueueManager(string queueManagerName, Int options, string channel,  
string connName);  
public MQQueueManager(string queueManagerName, string channel, string  
connName);  
public MQQueueManager(string queueManagerName, System.Collections.Hashtable  
properties);
```

Zgłasza MQException.

Tworzy połączenie z menedżerem kolejek. Wybierz między utworzeniem połączenia klienta lub połączenia z serwerem.

Podczas próby nawiązania połączenia z menedżerem kolejek konieczne jest sprawdzenie uprawnień (`inq`) w menedżerze kolejek. Próba nawiązania połączenia nie powiedzie się bez uzyskiwania informacji o uprawnieniach.

Połączenie klienta jest tworzone, jeśli spełniony jest jeden z następujących warunków:

1. `channel` lub `connName` są określone w konstruktorze.
2. `HostName`, `Port` lub `Channel` są określone w `properties`.
3. Określono `MQEnvironment.HostName`, `MQEnvironment.Port` lub `MQEnvironment.Channel`.

Wartości właściwości połączenia są domyślnie wyświetlane w podanej kolejności. Opcje `channel` i `connName` w konstruktorze mają pierwszeństwo przed wartościami właściwości w konstruktorze. Wartości właściwości konstruktora mają pierwszeństwo przed właściwościami `MQEnvironment`.

Nazwa hosta, nazwa kanału i port są zdefiniowane w klasie `MQEnvironment`.

`queueManagerName`

Nazwa menedżera kolejek lub grupy menedżerów kolejek, z którą ma zostać nawiązane połączenie.

Pomiń parametr lub pozostaw wartość `null` lub pole puste, aby dokonać wyboru domyślnego menedżera kolejek. Domyślne połączenie menedżera kolejek na serwerze jest domyślne dla menedżera kolejek na serwerze. Domyślne połączenie menedżera kolejek w połączeniu klienta jest związane z menedżerem kolejek, z którym jest połączony program nasłuchujący.

`options`

Określ opcje połączenia `MQCNO`. Wartości muszą mieć zastosowanie do typu nawiązanego połączenia. Jeśli na przykład zostaną określone następujące właściwości połączenia z serwerem dla połączenia klienckiego, zostanie zgłoszony `MQException`.

- `MQC.MQCNO_FASTPATH_BINDING`
- `MQC.MQCNO_STANDARD_BINDING`

`properties`

Parametr właściwości pobiera serię par klucz/wartość, które zastępują właściwości ustawione przez produkt `MQEnvironment`; patrz przykład [“Nadpisz właściwości `MQEnvironment`”](#) na stronie 1909. Następujące właściwości mogą zostać przesłonięte:

- `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

Nazwa kanału połączenia z serwerem

connName

Nazwa połączenia w formacie *HostName (Port)* .

Można podać listę *nazw hostów* i *portów* jako argument dla konstruktora `MQQueueManager` (`String queueManagerName`, `Hashtable properties`) przy użyciu właściwości `CONNECTION_NAME_PROPERTY`.

Na przykład:

```
ConnectionName = "fred.mq.com(2344),nick.mq.com(3746),tom.mq.com(4288)";
Hashtable Properties=new Hashtable();
properties.Add(MQC.CONNECTION_NAME_PROPERTY,ConnectionName);
MQQueueManager qmgr=new MQQueue Manager("qmgrname",properties);
```

Gdy podejmowana jest próba połączenia, lista nazw połączeń jest przetwarzana w kolejności. Jeśli próba nawiązania połączenia z pierwszą nazwą hosta i portem nie powiedzie się, zostanie podjęta próba nawiązania połączenia z drugą parą atrybutów. Klient powtarza ten proces, dopóki nie zostanie nawiązane połączenie, albo lista zostanie wyczerpana. Jeśli lista jest wyczerpana, do aplikacji klienckiej zwracany jest odpowiedni kod przyczyny i kod zakończenia.

Jeśli numer portu nie zostanie podany dla nazwy połączenia, port domyślny (skonfigurowany w produkcie `mqclient.ini`) jest używane.

Ustaw listę połączeń

Listę połączeń można ustawić, korzystając z następujących metod, gdy ustawione są opcje automatycznego ponownego połączenia klienta:

Ustaw listę połączeń za pomocą serwera MQSERVER

Listę połączeń można ustawić za pomocą wiersza komend.

W wierszu komend ustaw następującą komendę:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
```

Na przykład:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Jeśli połączenie zostało ustawione na serwerze MQSERVER, nie należy ustawiać go w aplikacji.

Jeśli lista połączeń zostanie ustawiona w aplikacji, aplikacja nadpisuje wszystko, co jest ustawione w zmiennej środowiskowej MQSERVER.

Ustaw listę połączeń za pomocą aplikacji

Listę połączeń w aplikacji można ustawić, określając nazwę hosta i właściwości portu.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```


Ustaw listę połączeń za pomocą `app.config`

`App.config` to plik XML, w którym określone są pary klucz-wartość.

Na liście połączeń określ

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
</app.Settings>
```

Na przykład:

```
<app.Settings>
<add key="Connection1" value="fred.mq.com(2966)"/>
<add key="Connection2" value="alex.mq.com(6533)"/>
</app.Settings>
```

Listę połączeń można bezpośrednio zmienić w pliku `app.config`.

Ustaw listę połączeń za pomocą `MQEnvironment`

Aby ustawić listę połączeń za pomocą `MQEnvironment`, należy użyć właściwości `ConnectionName`.

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),alex.mq.com(5211);
```

Właściwość `ConnectionName` powoduje nadpisanie właściwości nazwy hosta i portu ustawionych w `MQEnvironment`.

Tworzenie połączenia klienta

W poniższym przykładzie przedstawiono sposób tworzenia połączenia klienta z menedżerem kolejek. Połączenie z klientem można utworzyć, ustawiając zmienne produktu `MQEnvironment` przed utworzeniem nowego obiektu `MQQueueManager`.

```
MQEnvironment.Hostname = "fred.mq.com"; // host to connect to
MQEnvironment.Port      = 1414;          // port to connect to
                                     // If not explicitly set,
                                     // defaults to 1414
                                     // (the default 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");
```

Rysunek 43. Połączenie klienta

Nadpisz właściwości `MQEnvironment`

W poniższym przykładzie przedstawiono sposób tworzenia menedżera kolejek z jego identyfikatorem użytkownika i hasłem zdefiniowanym w tabeli mieszającej.

```

Hashtable properties = new Hashtable();

properties.Add( MQC.USER_ID_PROPERTY, "ExampleUserId" );
properties.Add( MQC.PASSWORD_PROPERTY, "ExamplePassword" );

try
{
    MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
}
catch (MQException mqe)
{
    System.Console.WriteLine("Connect failed with " + mqe.Message);
    return((int)mqe.Reason);
}

```

Rysunek 44. Nadpisywanie właściwości produktu MQEnvironment

Utwórz połączenie z możliwością ponownego połączenia

W poniższym przykładzie przedstawiono sposób automatycznego ponownego połączenia klienta z menedżerem kolejek.

```

Hashtable properties = new Hashtable(); // The queue manager name and the
// properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCN0_RECONNECT); // Options
// through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY, "fred.mq.com(4789),nick.mq.com(4790)"); // The list
// of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);

```

Rysunek 45. Automatyczne ponowne podłączanie klienta do menedżera kolejek

Klasa MQSubscription.NET

Użyj programu MQSubscription, aby zażądać, aby zachowane publikacje zostały wysłane do subskrybenta. MQSubscription to właściwość obiektu MQTopic otwartego na potrzeby subskrypcji.

Klasa

```

System.Object
├── IBM.WMQ.MQBase
│   └── IBM.WMQ.MQBaseObject
│       └── IBM.WMQ.MQManagedObject
│           └── IBM.WMQ.MQSubscription

```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- [“Właściwości” na stronie 1910](#)
- [“Metody” na stronie 1911](#)
- [“Konstruktory” na stronie 1911](#)

Właściwości

Dostęp do właściwości subskrypcji za pomocą klasy MQManagedObject; patrz [“Właściwości” na stronie 1867](#).

Metody

Uzyskaj dostęp do subskrypcji Inquire, Set i Get , korzystając z klasy MQManagedObject ; patrz [“Metody” na stronie 1868](#).

public int RequestPublicationUpdate(int options);

Zgłasza MQException.

Zażądaj zaktualizowanej publikacji dla bieżącego tematu. Jeśli menedżer kolejek ma zachowane publikacje dotyczące tematu, są one wysyłane do subskrybenta.

Przed wywołaniem programu RequestPublicationUpdateotwórz temat subskrypcji, aby uzyskać obiekt MQSubscription .

Zwykle należy otworzyć subskrypcję za pomocą opcji MQC.MQSO_PUBLICATIONS_ON_REQUEST . Jeśli w łańcuchu tematu nie ma znaków wieloznacznych, to w wyniku tego wywołania wysyłany jest tylko jedna publikacja. Jeśli łańcuch tematu zawiera znaki wieloznaczne, wiele publikacji może zostać wysłanych. Ta metoda zwraca liczbę zachowanych publikacji, które są wysyłane do kolejki subskrypcji. Nie ma gwarancji, że ta wiele publikacji zostanie odebranych, zwłaszcza jeśli są to komunikaty nietrwałe.

options

MQC.MQSRO_FAIL_IF QUIESCING

Metoda kończy się niepowodzeniem, jeśli menedżer kolejek znajduje się w stanie wygaszenia. W systemie z/OSdla aplikacji CICS lub IMS produkt MQC.MQSRO_FAIL_IF QUIESCING wymusza również niepowodzenie metody, jeśli połączenie jest w stanie wygaszonym.

MQC.MQSRO_NONE

Nie określono żadnych opcji.

Konstruktory

Brak konstruktora publicznego .

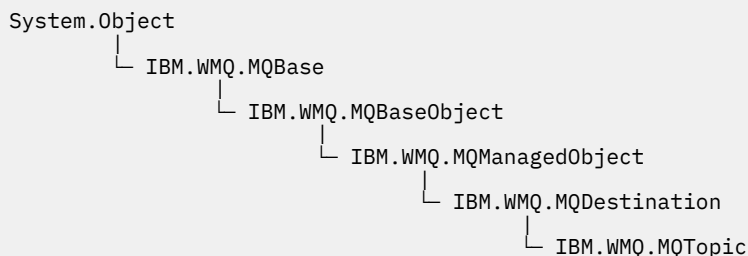
Obiekt MQSubscription jest zwracany we właściwości SubscriptionReference obiektu MQTopic , który jest otwierany na potrzeby subskrypcji,

Wywołaj metodę RequestPublicationUpdate . MQSubscription jest podklasą klasy MQManagedObject. Użyj odwołania, aby uzyskać dostęp do właściwości i metod produktu MQManagedObject.

Klasa MQTopic.NET

Produkt MQTopic służy do publikowania lub subskrybowania komunikatów w danym temacie, a także do tworzenia zapytań lub ustawiania atrybutów tematu. Utwórz obiekt MQTopic na potrzeby publikowania lub subskrybowania przy użyciu konstruktora lub metody MQQueueManager.AccessTopic .

Klasa



public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;

- [“Właściwości” na stronie 1912](#)

- [“Metody” na stronie 1912](#)
- [“Konstruktory” na stronie 1914](#)

Właściwości

Test dla `MQException` zgłaszanego podczas pobierania właściwości.

public Boolean IsDurable {get;}

Właściwość tylko do odczytu, która zwraca wartość `True` , jeśli subskrypcja jest trwała, lub `False` w przeciwnym razie. Jeśli temat został otwarty do publikacji, właściwość ta jest ignorowana i zawsze zwraca wartość `False`.

public Boolean IsManaged {get;}

Właściwość tylko do odczytu, która zwraca wartość `True` , jeśli subskrypcja jest zarządzana przez menedżer kolejek, lub `False` w przeciwnym razie. Jeśli temat został otwarty do publikacji, właściwość jest ignorowana i zawsze zwracana jest wartość `False`.

public Boolean IsSubscribed {get;}

Właściwość tylko do odczytu, która zwraca wartość `True` , jeśli temat został otwarty dla subskrypcji, i `False` , jeśli temat został otwarty do publikacji.

public MQSubscription SubscriptionReference {get;}

Właściwość tylko do odczytu, która zwraca obiekt `MQSubscription` powiązany z obiektem tematu otwartym na potrzeby subskrypcji. Odwołanie jest dostępne, jeśli użytkownik chce zmodyfikować opcje zamknięcia lub uruchomić dowolną z metod obiektów.

public MQDestination UnmanagedDestinationReference {get;}

Właściwość tylko do odczytu, która zwraca `MQQueue` powiązaną z subskrypcją niezarządzaną. Jest to miejsce docelowe określone podczas tworzenia obiektu tematu. Ta właściwość zwraca wartość `NULL` dla wszystkich obiektów tematów otwartych do publikacji lub z subskrypcją zarządzaną.

Metody

public void Put(MQMessage message);

public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);

Zgłasza wyjątek `MQException`.

Umożliwia opublikowanie komunikatu w temacie.

Modyfikacje obiektu `MQMessage` po zakończeniu wywołania `Put` nie mają wpływu na rzeczywisty komunikat w kolejce IBM MQ ani w temacie publikacji.

Produkt `Put` aktualizuje właściwości `MessageId` i `CorrelationId` obiektu `MQMessage` i nie powoduje czyszczenia danych komunikatu. Dalsze wywołania programu `Put` lub `Get` odnoszą się do zaktualizowanych informacji w obiekcie `MQMessage` . Na przykład w następującym fragmencie kodu pierwszy komunikat zawiera a i drugi ab.

```
msg.WriteString("a");
q.Put(msg, pmo);
msg.WriteString("b");
q.Put(msg, pmo);
```

message

Obiekt `MQMessage` zawierający dane deskryptora komunikatu i komunikat, który ma zostać wysłany. W wyniku tej metody deskryptor komunikatu może zostać zmieniony. Wartości w deskrytorze komunikatu natychmiast po zakończeniu tej metody to wartości, które zostały umieszczone w kolejce lub opublikowane w temacie.

Następujące kody przyczyny są zwracane do klienta z możliwością ponownego połączenia:

- `MQRC_CALL_INTERRUPTED` , jeśli połączenie zostało zerwane podczas wykonywania wywołania `Put` w trwałym komunikacie, a ponowne nawiązanie połączenia powiodło się.

- MQRC_NONE , jeśli połączenie jest pomyślne podczas uruchamiania wywołania Put w nietrwałym komunikacie (patrz sekcja Odtwarzanie aplikacji).

putMessageOptions

Opcje sterujące działaniem umieszczonym.

Jeśli produkt *putMessageOptions* nie jest określony, używana jest domyślna instancja produktu MQPutMessageOptions .

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQPMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: Aby uprościć i uzyskać wydajność, należy użyć obiektu MQQueueManager .Put , aby umieścić pojedynczy komunikat w kolejce. W tym celu należy mieć dla niego obiekt MQQueue .

```
public void Get(MQMessage message);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions);  
public void Get(MQMessage message, MQGetMessageOptions getMessageOptions, int  
MaxMsgSize);
```

Zgłasza wyjątek MQException.

Pobiera komunikat z tematu.

Ta metoda używa domyślnej instancji programu MQGetMessageOptions do wykonania operacji get. Używana opcja komunikatu to MQGMO_NOWAIT.

Jeśli operacja pobierania nie powiedzie się, obiekt MQMessage nie zostanie zmieniony. Jeśli operacja powiedzie się, deskryptor komunikatu i fragmenty danych komunikatu produktu MQMessage zostaną zastąpione przez deskryptor komunikatu i dane komunikatu z komunikatu przychodzącego.

Wszystkie wywołania programu IBM MQ z określonego MQQueueManager są synchroniczne. Z tego powodu, jeśli zostanie wykonane oczekiwanie, wszystkie inne wątki korzystające z tego samego serwera MQQueueManager są blokowane z kolejnych wywołań programu IBM MQ do czasu zakończenia operacji Get. Jeśli wymagane jest jednoczesne uzyskiwanie dostępu do produktu IBM MQ z wielu wątków, każdy wątek musi utworzyć własny obiekt MQQueueManager .

message

Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskrypcorze komunikatu są parametrami wejściowymi. Ważne jest, aby parametry wejściowe MessageId i CorrelationId zostały ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia, dla komunikatów odebranych w ramach produktu MQGM_SYNCPOINT.

getMessageOptions

Opcje sterujące działaniem get.

Użycie opcji MQC .MQGMO_CONVERT może spowodować wystąpienie wyjątku z kodem przyczyny MQC .MQRC_CONVERTED_STRING_TOO_BIG podczas przekształcania z jednobajtowych kodów znaków na kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie zostanie określony, użyta zostanie opcja MQGMO_NOWAIT.

Jeśli w kliencie z możliwością ponownego połączenia używana jest opcja MQGMO_LOGICAL_ORDER , zwracany jest kod przyczyny produktu MQRC_RECONNECT_INCOMPATIBLE .

MaxMsgSize

Największa wiadomość, którą ten obiekt komunikatu ma odebrać. Jeśli komunikat w kolejce jest większy niż ten rozmiar, występuje jedna z dwóch rzeczy:

- Jeśli flaga MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions , to komunikat jest wypełniany możliwie jak największą ilością danych

komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_ACCEPTED.

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny produktu MQRC_TRUNCATED_MSG_FAILED.

Jeśli parametr *MaxMsgSize* nie zostanie określony, zostanie pobrany cały komunikat.

Konstruktory

```
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

Dostęp do tematu w systemie *queueManager*.

Obiekty produktu MQTopic są ściśle powiązane z obiektami tematu administracyjnego, które są czasami nazywane obiektami tematów. Na wejściu element *topicObject* wskazuje na obiekt tematu administracyjnego. Konstruktor MQTopic uzyskuje łańcuch tematu z obiektu tematu i łączy go z *topicName* w celu utworzenia nazwy tematu. Albo oba obiekty *topicObject* lub *topicName* mogą mieć wartość NULL. Nazwa tematu zostanie dopasowana do drzewa tematów, a nazwa najbliższego zgodnego obiektu tematu administracyjnego jest zwracana w obiekcie *topicObject*.

Tematy powiązane z obiektem MQTopic są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez produkt *topicObject*. Drugi łańcuch tematu to *topicString*. Wynikowy łańcuch tematu powiązany z obiektem MQTopic może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwierany na potrzeby publikowania lub subskrybowania, można użyć metod MQTopic.Put do publikowania w tematach lub metod MQTopic.Get w celu otrzymywania publikacji na tematy. Aby opublikować ten sam temat i zasubskrybować ten sam temat, należy dwukrotnie uzyskać dostęp do tematu, raz na potrzeby publikowania i subskrypcji.

W przypadku utworzenia obiektu MQTopic dla subskrypcji, bez udostępniania obiektu MQDestination, zakłada się subskrypcję zarządzaną. Jeśli kolejka jest zaliczana jako obiekt MQDestination, zakłada się, że subskrypcja niezarządzana jest niezarządzana. Należy upewnić się, że ustawione opcje subskrypcji są spójne z subskrypcją zarządzaną lub niezarządzaną.

queueManager

Menedżer kolejek w celu uzyskania dostępu do tematu.

destination

destination jest instancją produktu MQQueue. Udostępniając produkt *destination*, produkt MQTopic jest otwierany jako subskrypcja niezarządzana. Publikacje dotyczące tego tematu są dostarczane do kolejki, do której dostęp jest uzyskiwany jako *destination*.

topicName

Łańcuch tematu, który jest drugą częścią nazwy tematu. *topicName* jest konkatelowany z łańcuchem tematu zdefiniowanym w administracyjnym obiekcie tematu *topicObject*. Parametr *topicName* można ustawić na wartość NULL, w którym to przypadku nazwa tematu jest definiowana przez łańcuch tematu w produkcie *topicObject*.

topicObject

Na wejściu *topicObject* jest nazwą obiektu tematu, który zawiera łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w produkcie *topicObject* jest konkatelowany z produktem *topicName*. Reguły konstruowania łańcuchów tematów są zdefiniowane w sekcji [łączenie łańcuchów tematów](#).

Na wyjściu program *topicObject* zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zbliżony w drzewie tematów do tematu określonego przez łańcuch tematu.

openAs

Przejdź do tematu, aby opublikować lub zasubskrybować. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

W tym celu należy połączyć opcje sterujące otwieraniem tematu w celu ich publikacji lub subskrypcji. Użyj stałych MQC.MQSO_*, aby uzyskać dostęp do tematu dotyczącego stałych subskrypcji i MQC.MQOO_*, aby uzyskać dostęp do tematu w celu opublikowania.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości razem lub połącz wartości opcji przy użyciu operatora OR bitowego.

alternateUserId

Podaj alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji wymaganej do zakończenia operacji. Wartość *alternateUserId* należy określić, jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY.

subscriptionName

subscriptionName jest wymagany, jeśli dostępne są opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER. W obu przypadkach produkt MQTopic jest niejawnie otwarty dla subskrypcji. Wyjątek jest zgłaszany, jeśli ustawiono MQC.MQSO_DURABLE, a subskrypcja istnieje, lub jeśli ustawiona jest wartość MQC.MQSO_ALTER, a subskrypcja nie istnieje.

properties

Ustaw dowolną z wymienionych właściwości subskrypcji przy użyciu tabeli mieszającej. Określone pozycje w tabeli mieszającej są aktualizowane z wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

```

public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int options, string alternateUserId, string subscriptionName);
public MQTopic MQQueueManager.AccessTopic(string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);

```

Uzyskaj dostęp do tematu dotyczącego tego menedżera kolejek.

Obiekty produktu `MQTopic` są ściśle powiązane z obiektami tematu administracyjnego, które są czasami nazywane obiektami tematów. Na wejściu element `topicObject` wskazuje na obiekt tematu administracyjnego. Konstruktor `MQTopic` uzyskuje łańcuch tematu z obiektu tematu i łączy go z `topicName` w celu utworzenia nazwy tematu. Albo oba obiekty `topicObject` lub `topicName` mogą mieć wartość `NULL`. Nazwa tematu zostanie dopasowana do drzewa tematów, a nazwa najbliższego zgodnego obiektu tematu administracyjnego jest zwracana w obiekcie `topicObject`.

Tematy powiązane z obiektem `MQTopic` są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez produkt `topicObject`. Drugi łańcuch tematu to `topicString`. Wynikowy łańcuch tematu powiązany z obiektem `MQTopic` może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwierany na potrzeby publikowania lub subskrybowania, można użyć metod `MQTopic.Put` do publikowania w tematach lub metod `MQTopic.Get` w celu otrzymywania publikacji na tematy. Aby opublikować ten sam temat i zasubskrybować ten sam temat, należy dwukrotnie uzyskać dostęp do tematu, raz na potrzeby publikowania i subskrypcji.

W przypadku utworzenia obiektu `MQTopic` dla subskrypcji, bez udostępniania obiektu `MQDestination`, zakłada się subskrypcję zarządzaną. Jeśli kolejka jest zaliczana jako obiekt `MQDestination`, zakłada się, że subskrypcja niezarządzana jest niezarządzana. Należy upewnić się, że ustawione opcje subskrypcji są spójne z subskrypcją zarządzaną lub niezarządzaną.

destination

`destination` jest instancją produktu `MQQueue`. Udostępniając produkt `destination`, produkt `MQTopic` jest otwierany jako subskrypcja niezarządzana. Publikacje dotyczące tego tematu są dostarczane do kolejki, do której dostęp jest uzyskiwany jako `destination`.

topicName

Łańcuch tematu, który jest drugą częścią nazwy tematu. `topicName` jest konkatelowany z łańcuchem tematu zdefiniowanym w administracyjnym obiekcie tematu `topicObject`. Parametr `topicName` można ustawić na wartość `NULL`, w którym to przypadku nazwa tematu jest definiowana przez łańcuch tematu w produkcie `topicObject`.

topicObject

Na wejściu `topicObject` jest nazwą obiektu tematu, który zawiera łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w produkcie `topicObject` jest konkatelowany z produktem `topicName`. Reguły konstruowania łańcuchów tematów są zdefiniowane w sekcji [łączenie łańcuchów tematów](#).

Na wyjściu program `topicObject` zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zbliżony w drzewie tematów do tematu określonego przez łańcuch tematu.

openAs

Przejdź do tematu, aby opublikować lub zasubskrybować. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

W tym celu należy połączyć opcje sterujące otwieraniem tematu w celu ich publikacji lub subskrypcji. Użyj stałych MQC.MQSO_*, aby uzyskać dostęp do tematu dotyczącego stałych subskrypcji i MQC.MQOO_*, aby uzyskać dostęp do tematu w celu opublikowania.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości razem lub połącz wartości opcji przy użyciu operatora OR bitowego.

alternateUserId

Podaj alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji wymaganej do zakończenia operacji. Wartość *alternateUserId* należy określić, jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY.

subscriptionName

subscriptionName jest wymagany, jeśli dostępne są opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER. W obu przypadkach produkt MQTopic jest niejawnie otwarty dla subskrypcji. Wyjątek jest zgłaszany, jeśli ustawiono MQC.MQSO_DURABLE, a subskrypcja istnieje, lub jeśli ustawiona jest wartość MQC.MQSO_ALTER, a subskrypcja nie istnieje.

properties

Ustaw dowolną z wymienionych właściwości subskrypcji przy użyciu tabeli mieszającej. Określone pozycje w tabeli mieszającej są aktualizowane z wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

Interfejs IMQObjectTrigger.NET

Zaimplementuj program IMQObjectTrigger w celu przetwarzania komunikatów przekazywanych przez monitor **runmqdmn.NET**.

Interfejs

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

W zależności od tego, czy element sterujący punktu synchronizacji jest określony w komendzie **runmqdmn**, komunikat jest usuwany z kolejki przed lub po powrocie z metody **Execute**.

Metody

void Execute (MQQueueManager queueManager, MQQueue queue, MQMessage message, string param);

queueManager

Menedżer kolejek udostępniający kolejkę, która jest monitorowana.

queue

Monitorowana kolejka.

message

Komunikat odczytany z kolejki.

param

Dane przekazane z UserParameter.

Interfejs MQC.NET

Należy odwołać się do stałej MQI , poprzedzając stałą nazwę za pomocą MQC . . MQC definiuje wszystkie stałe używane przez MQI.

Interfejs

```
System.Object
└─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Przykład

```
MQQueue queue;  
queue.closeOptions = MQC.MQCO_DELETE;
```

Identyfikatory zestawów znaków dla aplikacji produktu .NET

Opisy zestawów znaków, które można wybrać, aby zakodować komunikaty programu .NET IBM MQ

Zestaw znaków	Opis
37	ibm037
437	ibm437 /PC Oryginał
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

Zestaw znaków	Opis
297	ibm297
420	ibm420
424	ibm424
737	ibm737 /PC grecki
775	ibm775 /PC Bałtycki
813	iso-8859-7 /greek/ ibm813
838	ibm838
850	ibm850 /PC Latin 1
852	ibm852 /PC Latin 2
855	ibm855 /PC Cyrillic
856	ibm856
857	ibm857 /PC turecki
860	ibm860 /PC portugalski
861	ibm861 /PC islandzki
862	ibm862 /PC hebrajski
863	ibm863 /PC kanadyjski francuski
864	ibm864 /PC arabski
865	ibm865 /PC Nordic
866	ibm866 /PC rosyjski
868	ibm868
869	ibm869 /PC Modern Greek
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 /cyrillic/ ibm915
916	iso-8859-8 /hebrew/ ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930

Zestaw znaków	Opis
932	PC japoński
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 /Big 5-chiński tradycyjny
954	EUCJIS
964	ibm964 /CNS 11643 Traditional Chinese (chiński tradycyjny)
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 /arabic/ ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latin 2
1251	Windows cyrylica
1252	Windows Latin 1
1253	Windows grecki
1254	Windows turecki
1255	Windows hebrajski
1256	Windows arabski
1257	Windows bałtycki
1258	Windows wietnamski
1381	ibm1381
1383	ibm1383
2022	JIS

Zestaw znaków	Opis
5601	ksc-5601 koreański
33722	ibm33722

Klasy C++ w programie IBM MQ

Klasy języka C++ programu IBM MQ hermetyzują interfejs kolejki komunikatów produktu IBM MQ (MQI). Dostępny jest pojedynczy plik nagłówkowy C++ **imqi.hpp**, który obejmuje wszystkie te klasy.

Dla każdej klasy wyświetlane są następujące informacje:

Diagram hierarchii klas

Diagram klas przedstawiający klasę w relacji dziedziczenia z jej bezpośrednimi klasami nadrzędnymi (jeśli istnieją).

Inne istotne klasy

Odsyłacze dokumentów do innych odpowiednich klas, takich jak klasy macierzyste, oraz klas obiektów używanych w sygnaturach metod.

Atrybuty obiektu

Atrybuty klasy. Są to oprócz atrybutów zdefiniowanych dla klas nadrzędnych. Wiele atrybutów odzwierciedla elementy struktury danych produktu IBM MQ (patrz [“Skorowidz języka C++ i MQI” na stronie 1922](#)). Szczegółowe opisy znajdują się w sekcji [“Atrybuty obiektów” na stronie 810](#).

Konstruktory

Sygnatury metod specjalnych używanych do tworzenia obiektu klasy.

Metody obiektów (publiczne)

Podpisy metod, które wymagają instancji klasy dla ich operacji i nie mają ograniczeń użycia.

W przypadku, gdy ma to zastosowanie, przedstawione są również następujące informacje:

Metody klasy (publiczne)

Podpisy metod, które nie wymagają instancji klasy dla ich operacji i nie mają ograniczeń użycia.

Metody przeciążone (klasy macierzyste)

Podpisy tych metod wirtualnych, które są zdefiniowane w klasach macierzystych, ale wykazują różne, polimorficzne, zachowania dla tej klasy.

Metody obiektów (chronione)

Podpisy metod, które wymagają instancji klasy dla ich działania i są zarezerwowane do użycia przez implementacje klas pochodnych. Ta sekcja jest interesowana tylko dla programów piszących klasy, w przeciwieństwie do użytkowników klasy.

Dane obiektu (chronione)

Szczegóły implementacji dla danych instancji obiektu dostępnych dla implementacji klas pochodnych. Ta sekcja jest interesowana tylko dla programów piszących klasy, w przeciwieństwie do użytkowników klasy.

Kody przyczyny

Wartości MQRC_* (patrz sekcja [Zakończenie interfejsu API i kody przyczyny](#)). można się spodziewać po tych metodach, które się nie powiodły. Wyczerpujący wykaz kodów przyczyny, które mogą wystąpić dla obiektu klasy, można znaleźć w dokumentacji klasy nadrzędnej. Udokumentowana lista kodów przyczyn dla klasy nie zawiera kodów przyczyny dla klas nadrzędnych.

Uwaga:

1. Obiekty z tych klas nie są wątkowo bezpieczne. Zapewnia to optymalną wydajność, ale dbanie o to, aby nie uzyskiwać dostępu do żadnego obiektu z więcej niż jednego wątku.
2. Zaleca się, aby dla programu wielowątkowego dla każdego wątku używany był osobny obiekt `ImqQueueManager`. Każdy obiekt menedżera musi mieć własną niezależną kolekcję innych obiektów, zapewniając, że obiekty w różnych wątkach są odizolowane od siebie.

Dostępne są następujące klasy:

- [“ImqAuthentication-rejestrowanie klasy C++” na stronie 1938](#)
- [“Klasa ImqBinary C++” na stronie 1940](#)
- [“Klasa ImqCache C++” na stronie 1942](#)
- [“Klasa języka C++ ImqChannel” na stronie 1945](#)
- [“ImqCICSBridgeHeader C++” na stronie 1950](#)
- [“Klasa języka C++ ImqDeadLetterHeader” na stronie 1957](#)
- [“Klasa ImqDistribution-lista C++” na stronie 1959](#)
- [“Klasa języka C++ ImqError” na stronie 1961](#)
- [“ImqGetMessageOptions klasa C++” na stronie 1962](#)
- [“Klasa języka C++ ImqHeader” na stronie 1965](#)
- [“ImqIMSBridgeHeader C++” na stronie 1967](#)
- [“Klasa ImqItem C++” na stronie 1970](#)
- [“Klasa języka C++ ImqMessage” na stronie 1971](#)
- [“Klasa ImqMessageTracker C++” na stronie 1978](#)
- [“Klasa ImqNamelist C++” na stronie 1981](#)
- [“Klasa języka C++ ImqObject” na stronie 1983](#)
- [“Klasa ImqProcess C++” na stronie 1989](#)
- [“Klasa języka C++ ImqPutMessageOptions” na stronie 1990](#)
- [“Klasa ImqQueue C++” na stronie 1992](#)
- [“Klasa C++ programu ImqQueueManager” na stronie 2003](#)
- [“Klasa ImqReferencenagłówka C++” na stronie 2019](#)
- [“Klasa ImqString C++” na stronie 2022](#)
- [“Klasa ImqTrigger C++” na stronie 2027](#)
- [“Klasa ImqWorknagłówka C++” na stronie 2030](#)

Skorowidz języka C++ i MQI

Ta kolekcja tematów zawiera informacje dotyczące języka C++ na potrzeby interfejsu MQI.

Zapoznaj się z tą informacją razem z produktem [“Typy danych używane w interfejsie MQI” na stronie 237](#).

Ta tabela dotyczy struktur danych MQI dla klas C++ i plików włączanych. Poniższe tematy zawierają informacje uzupełniające dla każdej klasy C++. Te odniesienia dotyczą korzystania z bazowych interfejsów proceduralnych IBM MQ. Klasy ImqBinary, ImqDistributionList i ImqString nie mają atrybutów, które należą do tej kategorii i są wykluczane.

<i>Tabela 221. Struktura danych, klasa i odwołanie do pliku włączeń/zbioru</i>		
Struktura danych	Klasa	Plik włączany
MQAIR	Rekord ImqAuthentication	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH,	ImqCICSBridgeHeader	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	Lista ImqDistribution	imqdst.hpp
	ImqError	imqerr.hpp

Tabela 221. Struktura danych, klasa i odwołanie do pliku włączeń/zbioru (kontynuacja)

Struktura danych	Klasa	Plik włączany
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH.	ImqIMSBridgeHeader	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	ImqMessageTracker	imqmtr.hpp
	ImqNamelist	imqnml.hpp
MQOD, MQRR	ImqObject	imqobj.hpp
MQPMO, MQPMR, MQRR	ImqPutMessageOptions	imqpmo.hpp
	ImqProcess	imqpro.hpp
	ImqQueue	imqque.hpp
MQBO, MQCNO, MQCSP	Menedżer ImqQueue	imqmgr.hpp
MQRMH	Nagłówek ImqReference	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIIH	Nagłówek ImqWork	imqwih.hpp

ImqAuthentication-odniesienie do rekordu

Odwołanie krzyżowe atrybutów, struktur danych, pól i wywołań klasy ImqAuthenticationRecord C++.

Atrybut	Struktura danych	Pole	Wywołanie
Typ połączenia	MQAIR	AuthInfoConnName	MQCONN
Hasło	MQAIR	Hasło_LDAPPassword	MQCONN
typ	MQAIR	Typ AuthInfo	MQCONN
nazwa użytkownika,	MQAIR	LDAPUserNamePtr	MQCONN
	MQAIR	Przesunięcie LDAPUserName	MQCONN
	MQAIR	Długość elementu LDAPUserName	MQCONN

ImqCache -referencja

Odwołanie do atrybutów i wywołań klasy ImqCache C + +.

Atrybut	Wywołanie
bufor automatyczny	MQGET

Atrybut	Wywołanie
długość buforu	MQGET
wskaźnik buforu	MQGET, MQPUT
Długość danych	MQGET
Pozycja danych	MQGET
wskaźnik danych	MQGET
długość komunikatu	MQGET, MQPUT

ImqChannel -referencja

Odwwołanie do atrybutów, struktur danych, pól i wywołań dla klasy ImqChannel C + +.

Atrybut	Struktura danych	Pole	Wywołanie
wsadowe bicie serca	MQCD	BatchHeartbeat	MQCONN
nazwa kanału	MQCD	ChannelName	MQCONN
Typ połączenia	MQCD	ConnectionName	MQCONN
	MQCD	ShortConnectionNazwa	MQCONN
Kompresja nagłówka	MQCD	Lista HdrComp	MQCONN
interwał pulsu	MQCD	HeartbeatInterval	MQCONN
Interwał sprawdzania połączenia	MQCD	Przedział czasu KeepAlive	MQCONN
Adres lokalny	MQCD	LocalAddress	MQCONN
Maksymalna długość komunikatu	MQCD	MaxMsgDługość	MQCONN
Kompresja komunikatu	MQCD	Lista MsgComp	MQCONN
Nazwa trybu	MQCD	ModeName	MQCONN
Hasło	MQCD	Hasło	MQCONN
liczba wyjść odbierania	MQCD		MQCONN
nazwy wyjścia odbierania	MQCD	ReceiveExit	MQCONN
	MQCD	ReceiveExitsZdefiniowano	MQCONN
	MQCD	ReceiveExitPtr	MQCONN
otrzymywanie danych użytkownika	MQCD	Dane ReceiveUser	MQCONN
	MQCD	ReceiveUserDataPtr	MQCONN
Nazwa wyjścia zabezpieczeń	MQCD	SecurityExit	MQCONN
dane użytkownika zabezpieczeń	MQCD	Dane SecurityUser	MQCONN
Liczba wyjść wysyłania	MQCD		MQCONN
nazwy wyjścia wysyłania	MQCD	SendExit	MQCONN
	MQCD	SendExitsZdefiniowano	MQCONN
	MQCD	SendExitPtr	MQCONN
wysyłanie danych użytkownika	MQCD	Dane SendUser	MQCONN

Atrybut	Struktura danych	Pole	Wywołanie
	MQCD	SendUserDataPtr	MQCONN
CipherSpec SSL	MQCD	Specyfikacja sslCipher	MQCONN
Typ uwierzytelniania klienta SSL	MQCD	Uwierzytelnianie sslClient	MQCONN
Nazwa węzła sieci SSL	MQCD	SSLPEERNAME	MQCONN
Nazwa programu transakcyjnego	MQCD	TpName	MQCONN
Typ transportu	MQCD	TransportType	MQCONN
ID użytkownika	MQCD	UserIdentifier	MQCONN

ImqCICSBridgeHeader -referencja

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqCICSBridgeHeader C + +.

Atrybut	Struktura danych	Pole
kod abend mostu	MQCIH,	AbendCode
ADS, deskryptor	MQCIH,	AdsDescriptor
identyfikator uwagi	MQCIH,	AttentionId
element uwierzytelniający	MQCIH,	Authenticator
kod zakończenia mostu	MQCIH,	Kod BridgeCompletion
przesunięcie błędu mostu	MQCIH,	ErrorOffset
kod przyczyny mostu	MQCIH,	BridgeReason
kod anulowania mostu	MQCIH,	CancelCode
zadanie konwersacyjne	MQCIH,	ConversationalTask
pozycja kursora	MQCIH,	CursorPosition
znacznik narzędzia	MQCIH,	Udogodnienia
czas przechowywania	MQCIH,	Czas przechowywania FacilityKeep
Obiekt podobny	MQCIH,	FacilityLike
function (funkcja)	MQCIH,	Funkcja
okres oczekiwania na pobranie	MQCIH,	Przedział czasu GetWait
Typ odsyłacza	MQCIH,	LinkType
następny identyfikator transakcji	MQCIH,	Identyfikator NextTransaction
długość danych wyjściowych	MQCIH,	Długość OutputData
format odpowiedzi	MQCIH,	Format ReplyTo
kod powrotu mostu	MQCIH,	ReturnCode
kod początkowy	MQCIH,	StartCode
status zakończenia zadania	MQCIH,	Status TaskEnd
Identyfikator transakcji	MQCIH,	TransactionId
uow, sterowanie	MQCIH,	UowControl

Atrybut	Struktura danych	Pole
wersja	MQCIH,	Wersja

ImqDeadLetterHeader odniesień

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqDeadLetterHeader C++.

Atrybut	Struktura danych	Pole
kod przyczyny niedostarczenia	MQDLH	Przyczyna
Nazwa menedżera kolejek docelowych	MQDLH	Docelowy_menedżer_kolejek
nazwa kolejki docelowej	MQDLH	DestQName
Nazwa aplikacji wstawiającej	MQDLH	Nazwa_aplikacji_wstawiającej
Typ aplikacji wstawiającej	MQDLH	Typ_aplikacji_wstawiającej
Data wstawienia	MQDLH	PutDate
Czas wstawienia	MQDLH	PutTime

Odwołanie krzyżowe ImqError

Odwołanie do atrybutów i wywołań klasy C++ ImqError .

Atrybut	Wywołanie
kod zakończenia	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET
reason code (kod przyczyny)	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

ImqGetMessageOptions -odwołanie wzajemne

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqGetMessageOptions C++.

Atrybut	Struktura danych	Pole
Status grupy	MQGMO	GroupStatus
opcje dopasowywania	MQGMO	MatchOptions
znacznik komunikatu	MQGMO	MessageToken
opcje.	MQGMO	Opcje
rozstrzygnięta nazwa kolejki	MQGMO	ResolvedQName
zwrócona długość	MQGMO	ReturnedLength
segmentacja	MQGMO	Segmentacja
status segmentu	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
Uczestnictwo w synchronizacji	MQGMO	Opcje
Interwał oczekiwania	MQGMO	WaitInterval

Odwołanie krzyżowe ImqHeader

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqHeader C++.

Atrybut	Struktura danych	Pole
zestaw znaków	MQDLH, MQIIH	CodedCharSetId
encoding	MQDLH, MQIIH	Kodowanie
format	MQDLH, MQIIH	Format
flagi nagłówka	MQIIH, MQRMH	Flagi

Odwołanie krzyżowe imqIMSBridgeHeader

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord C + +.

Atrybut	Struktura danych	Pole
element uwierzytelniający	MQIIH.	Authenticator
tryb zatwierdzania	MQIIH.	CommitMode
nadpisanie terminalu logicznego	MQIIH.	LTermOverride
nazwa odwzorowania usług formatu komunikatów	MQIIH.	MFSMapName
format odpowiedzi	MQIIH.	Format ReplyTo
zasięg zabezpieczeń	MQIIH.	SecurityScope
identyfikator instancji transakcji	MQIIH.	Identyfikator TranInstance
Stan transakcji	MQIIH.	TranState

Odwołanie krzyżowe ImqItem

Odwołanie do atrybutów i wywołań klasy ImqItem C + +.

Atrybut	Wywołanie
identyfikator struktury	MQGET

Odwołanie krzyżowe ImqMessage

Odwołanie do atrybutów, struktur danych, pól i wywołań klasy C++ ImqMessage .

Atrybut	Struktura danych	Pole	Wywołanie
Informacje identyfikujące aplikację	MQMD	Dane_tożsamości_aplikacji	
Dane_pochodzenia_aplikacji	MQMD	Dane_pochodzenia_aplikacji	
Licznik wycofań	MQMD	BackoutCount	
zestaw znaków	MQMD	CodedCharSetId	
encoding	MQMD	Kodowanie	
Utrata ważności	MQMD	Utrata ważności	
format	MQMD	Format	
Flagi komunikatu	MQMD	MsgFlags	

Atrybut	Struktura danych	Pole	Wywołanie
typ komunikatu	MQMD	MsgType	
Przesunięcie	MQMD	Depozycja	
Pierwotna długość	MQMD	OriginalLength	
trwałość	MQMD	Trwałość	
priorytet	MQMD	Priorytet	
Nazwa aplikacji wstawiającej	MQMD	Nazwa_aplikacji_wstawiającej	
Typ aplikacji wstawiającej	MQMD	Typ_aplikacji_wstawiającej	
Data wstawienia	MQMD	PutDate	
Czas wstawienia	MQMD	PutTime	
odpowiedź-na nazwę menedżera kolejek	MQMD	ReplyToQMgr	
Nazwa kolejki odpowiedzi	MQMD	ReplyToQ	
raport	MQMD	Raport	
numer kolejny	MQMD	Numer_kolejny_komunikatu	
całkowita długość komunikatu		DataLength	MQGET
ID użytkownika	MQMD	UserIdentifier	

ImqMessage-odniesienie do programu śledzący

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqMessageTracker C++.

Atrybut	Struktura danych	Pole
Token rozliczania	MQMD	AccountingToken
Identyfikator korelacji	MQMD	CorrelId
Feedback	MQMD	Opinie
Identyfikator grupy	MQMD	GroupId
Identyfikator komunikatu	MQMD	MsgId

Odwołanie do tabeli ImqNamelist

Odwołanie do atrybutów, zapytań i wywołań dla klasy ImqNamelist w języku C++.

Atrybut	Zapytanie	Wywołanie
Liczba nazw	LICZBA NAZW MQIA_NAME_COUNT	MQINQ
Nazwa listy nazw	NAZWA_LISTY_MQC	MQINQ

Odwołanie krzyżowe ImqObject

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań klasy C++ ImqObject .

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
Data zmiany			MQCA_ALTERATION_DATE	MQINQ
Godzina zmiany			MQCA_ALTERATION_TIME	MQINQ
Alternatywne ID użytkownika	MQOD	Identyfikator AlternateUser		
alternatywny identyfikator zabezpieczeń				
opcje zamknięcia				MQCLOSE
opis			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nazwa	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opcje otwarcia				MQOPEN
status otwarcia				MQOPEN, MQCLOSE
identyfikator menedżera kolejek	identyfikator menedżera kolejek		IDENTYFIKATOR_MENEDŻERA_KOLEJEK MQCA_Q_MGR_IDENTIFIER	MQINQ

ImqProcess -referencja

Odwołanie do atrybutów, zapytań i wywołań dla klasy ImqAuthenticationRecord C + +.

Atrybut	Zapytanie	Wywołanie
Identyfikator aplikacji	MQCA_APPL_ID	MQINQ
Typ aplikacji	MQIA_APPL_TYPE	MQINQ
Dane środowiska	MQCA_ENV_DATA	MQINQ
Dane użytkownika	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions -referencja

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord C + +.

Tabela 222. Odwołanie krzyżowe ImqPutMessageOptions		
Atrybut	Struktura danych	Pole
odwołanie do kontekstu	MQPMO	Kontekst
	MQPMO	Liczba InvalidDest
	MQPMO	Liczba KnownDest
opcje.	MQPMO	Opcje
pola rekordu	MQPMO	PutMsgRecFields
rozstrzygnięta nazwa menedżera kolejek	MQPMO	Nazwa ResolvedQMgr

Tabela 222. Odwołanie krzyżowe ImqPutMessageOptions (kontynuacja)		
Atrybut	Struktura danych	Pole
rozstrzygnięta nazwa kolejki	MQPMO	ResolvedQName
	MQPMO	Limit czasu
	MQPMO	Liczba UnknownDest
Uczestnictwo w synchronizacji	MQPMO	Opcje

ImqQueue -referencja

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań klasy C++ ImqQueue .

Tabela 223. Odwołanie wzajemne ImqQueue				
Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
nazwa kolejki wycofanych komunikatów			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
Próg wycofania			MQIA_BACKOUT_THRESHOLD	MQINQ
podstawowa nazwa kolejki			MQCA_BASE_Q_NAME	MQINQ
nazwa klastra			MQCA_NAZWA_KLASTRA	MQINQ
Nazwa listy nazw klastrów			MQCA_CLUSTER_NAMELIST,	MQINQ
Klasyfikacja obciążenia klastrów			MQIA_CLWL_Q_RANK	MQINQ
Priorytet obciążenia klastrów			MQIA_CLWL_Q_PRIORITY	MQINQ
Kolejka użycia obciążenia klastra			MQIA_CLWL_USEQ	MQINQ
data utworzenia			MQCA_CREATION_DATE	MQINQ
Godzina utworzenia			MQCA_CREATION_TIME	MQINQ
Bieżące zapełnienie			MQIA_CURRENT_Q_DEPTH	MQINQ
powiązanie domyślne			MQIA_DEF_BIND	MQINQ
Domyślna opcja otwarcia wejścia			MQIA_DEF_INPUT_OPEN_OPTION,	MQINQ
Trwałość domyślna			MQIA_DEF_PERSISTENCE	MQINQ
Domyślny priorytet			MQIA_DEF_PRIORITY,	MQINQ
Typ definicji			TYP_definicji_MQIA_MQS	MQINQ
zdarzenie o dużej głębokości			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
wysoki limit głębokości			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ

Tabela 223. Odwołanie wzajemne ImqQueue (kontynuacja)				
Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
zdarzenie o niskiej głębokości			MQIA_Q_DEPTH_LOW_EVENT	MQINQ
limit głębokości			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
zdarzenie maksymalnego zapętnienia			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
listy dystrybucyjne			MQIA_DIST_LISTS	MQINQ, MQSET
nazwa kolejki dynamicznej	MQOD	DynamicQName		
Zapisane wycofane komunikaty			MQIA_HARDEN_GET_BACKOUT	MQINQ
Typ indeksu			TYP_INDEKSU MQIA_INDEX_TYPE	MQINQ
inhibit get			MQIA_INHIBIT_GET	MQINQ, MQSET
zablokuj wstawianie			MQIA_INHIBIT_PUT	MQINQ, MQSET
Nazwa kolejki inicjacji			MQCA_INITIATION_Q_NAME	MQINQ
Zapełnienie maksymalne			MQIA_MAX_Q_DEPTH	MQINQ
Maksymalna długość komunikatu			MAKSYMALNA_DŁUGOŚĆ_WYWOŁANIA	MQINQ
Kolejność dostarczania komunikatów			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
kolejka rozproszona				
Klasa komunikatów nietrwałych			KLASA MQIA_NPM_CLASS	MQINQ
Liczba otwartych wejść			MQIA_OPEN_INPUT_COUNT,	MQINQ
Liczba otwartych wyjść			MQIA_OPEN_OUTPUT_COUNT	MQINQ
poprzednia kolejka rozproszona				
Nazwa procesu			NAZWA PROCESU MQCA_PROCESS_NAME	MQINQ
Rozliczanie kolejek			MQIA_ACCOUNTING_Q	MQINQ
Nazwa menedżera kolejek	MQOD	Nazwa ObjectQMgr		
Monitorowanie kolejek			MQIA_MONITORING_Q	MQINQ

Tabela 223. Odwołanie wzajemne ImqQueue (kontynuacja)				
Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
Statystyka kolejek			MQIA_STATISTICS_Q	MQINQ
Typ kolejki			TYP_Q_MQIA_MQ	MQINQ
Nazwa zdalnego menedżera kolejek			MQCA_REMOTE_Q_MGR_NAME,	MQINQ
Nazwa zdalnej kolejki			MQCA_REMOTE_Q_NAME	MQINQ
rozstrzygnięta nazwa menedżera kolejek	MQOD	Nazwa ResolvedQMgr		
rozstrzygnięta nazwa kolejki	MQOD	ResolvedQName		
Interwał przechowywania			MQIA_RETENTION_INTERVAL	MQINQ
zasięg			MQIA_SCOPE	MQINQ
interwał usług			MQIA_Q_SERVICE_INTERVAL	MQINQ
zdarzenie interwału usług			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
Możliwość współużytkowania			Funkcja MQIA_SHAREABILITY	MQINQ
klasa pamięci masowej			MQCA_STORAGE_CLASS,	MQINQ
Nazwa kolejki transmisji			MQCA_XMIT_Q_NAME	MQINQ
Kontrola wyzwalacza			MQIA_TRIGGER_CONTROL	MQINQ, MQSET
Dane wyzwalacza			MQCA_TRIGGER_DATA,	MQINQ, MQSET
Wyzwalacz uruchamiany wypełnieniem			MQIA_TRIGGER_DEPTH	MQINQ, MQSET
Priorytet komunikatu wyzwalacza			MQIA_TRIGGER_MSG_PRIORITY,	MQINQ, MQSET
typ wyzwalacza			MQIA_TRIGGER_TYPE	MQINQ, MQSET
użycie			SKŁADNIA MQIA_USAGE	MQINQ

Odwołanie do menedżera ImqQueueManager

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań dla klasy ImqQueueManager + +.

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
nadpisywanie połączeń rozliczania			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
Interwał rozliczania			MQIA_ACCOUNTING_INTERVAL	MQINQ
Zapis aktywności			MQIA_ACTIVITY_RECORDING	MQINQ
Sprawdzenie dołączenia nowego MCA			MQIA_ADOPTNEWMCA_CHECK	MQINQ
Typ dołączenia nowego MCA			MQIA_ADOPTNEWMCA_TYPE	MQINQ
Typ uwierzytelniania	Protokół MQCSP	AuthenticationType		MQCONN
zdarzenie uprawnienia			MQIA_AUTHORITY_EVENT,	MQINQ
opcje rozpoczęcia	MQBO	Opcje		MQBEGIN
zdarzenie mostu			MQIA_BRIDGE_EVENT	MQINQ
Automatyczna definicja kanału			MQIA_CHANNEL_AUTO_DEF	MQINQ
zdarzenie automatycznego definiowania kanału			MQIA_CHANNEL_AUTO_EVENT	MQIA
Wyjście automatycznej definicji kanału			MQIA_CHANNEL_AUTO_EXIT	MQIA
zdarzenie kanału			MQIA_CHANNEL_EVENT	MQINQ
Adaptory inicjatora kanału			MQIA_CHINIT_ADAPTERS	MQINQ
Kontrola inicjatora kanału			MQIA_CHINIT_CONTROL	MQINQ
Programy rozsyłające inicjatora kanału			MQIA_CHINIT_DISPATCHERS	MQINQ
Autostart śledzenia inicjatora kanału			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Wielkość tabeli śledzenia inicjatora kanału			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
Monitorowanie kanałów			MQIA_MONITORING_CHANNEL	MQINQ

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
odwołanie do kanału	MQCD	ChannelType		MQCONN
Statystyka kanałów			Kanał MQIA_STATISTICS_CHANNEL	MQINQ
zestaw znaków			MQIA_CODED_CHAR_SET_ID	MQINQ
Monitorowanie nadawcy klastrów			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Statystyka nadawcy klastrów			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Dane obciążenia klastra			MQCA_CLUSTER_WORKLOAD_DATA,	MQINQ
Wyjście obciążenia klastra			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
Długość obciążenia klastra			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
obciążenie klastra mru			MQIA_CLWL_MRU_CHANNELS	MQINQ
Kolejka użycia obciążenia klastra			MQIA_CLWL_USEQ	MQINQ
zdarzenie komendy			MQIA_COMMAND_EVENT	MQINQ
nazwa kolejki wejściowej komendy			MQCA_KOMEND_WEJŚCIOWY_Q_NAME	MQINQ
poziom komendy			MQIA_COMMAND_LEVEL	MQINQ
Kontrola serwera komend			MQIA_CMD_SERVER_CONTROL	MQINQ
Opcje połączenia	MQCNO	Opcje		MQCONN, MQCONN
Identyfikator połączenia	MQCNO	ConnectionId		MQCONN
status połączenia				MQCONN, MQCONN, MQDISC
Znacznik połączenia	MQCD	ConnTag		MQCONN
Sprzęt szyfrujący	MQSCO	CryptoHardware		MQCONN
nazwa kolejki niedostarczonych komunikatów			MQCA_DEAD_LETTER_Q_NAME	MQINQ
domyślna nazwa kolejki transmisji			MQCA_DEF_XMIT_Q_NAME	MQINQ
listy dystrybucyjne			MQIA_DIST_LISTS	MQINQ

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
grupa dns			MQCA_DNS_GROUP	MQINQ
dns wlm			MQIA_DNS_WLM	MQINQ
pierwszy rekord uwierzytelniania	MQSCO	AuthInfoRecOffset		MQCONN
	MQSCO	AuthInfoRecPtr		MQCONN
Hamuj zdarzenie			Zdarzenie MQIA_INHIBIT_EVENT	MQINQ
Wersja adresu IP			MQIA_IP_ADDRESS_VERSION	MQINQ
repozytorium kluczy	MQSCO	KeyRepository		MQCONN
licznik resetowania klucza	MQSCO	Licznik KeyReset		MQCONN
Zegar nasłuchiwania			MQIA_LISTENER_TIMER	MQINQ
zdarzenie lokalne			MQIA_LOCAL_EVENT,	MQINQ
zdarzenie programu rejestrującego			MQIA_LOGGER_EVENT,	MQINQ
Nazwa grupy LU			MQCA_LU_NAZWA_GRUPY	MQINQ
Nazwa LU			MQCA_LU_NAME	MQINQ
Przyrostek ramienia lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
Kanały lu62			MQIA_LU62_CHANNELS	MQINQ
maksymalna liczba aktywnych kanałów			MQIA_ACTIVE_CHANNELS	MQINQ
Maksimum kanałów			MQIA_MAX_CHANNELS	MQINQ
maksymalne uchwyt			MQIA_MAX_UCHWYTY	MQINQ
Maksymalna długość komunikatu			MAKSYMALNA_DŁUGOŚĆ_WYWOŁANIA	MQINQ
maksymalny priorytet			MQIA_MAX_PRIORITY	MQINQ
Maks. liczba niezatw. kom.			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
Rozliczanie MQI			MQIA_ACCOUNTING_MQI,	MQINQ
Statystyka MQI			MQIA_STATISTICS_MQI	MQINQ
maksymalny port wychodzący			MQIA_OUTBOUND_PORT_MAX	MQINQ

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
minimum portu wychodzącego			MQIA_OUTBOUND_PORT_MIN	MQINQ
Hasło	Protokół MQCSP	CSPPasswordPtr		MQCONN
	Protokół MQCSP	CSPPasswordOffset		MQCONN
	Protokół MQCSP	CSPPasswordLength		MQCONN
zdarzenie dotyczące wydajności			MQIA_PERFORMANCE_EVENT	MQINQ
platforma			PLATFORMA mqia_platforma	MQINQ
Rozliczanie kolejek			MQIA_ACCOUNTING_Q	MQINQ
Monitorowanie kolejek			MQIA_MONITORING_Q	MQINQ
Statystyka kolejek			MQIA_STATISTICS_Q	MQINQ
Limit czasu odbierania			MQIA_RECEIVE_TIMEOUT	MQINQ
minimum limitu czasu odbierania			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Typ limitu czasu odbierania			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
zdarzenie zdalne			MQIA_REMOTE_EVENT,	MQINQ
Nazwa repozytorium			NAZWA_REPOZYTORIUM_MQCA	MQINQ
Lista nazw repozytorium			MQCA_REPOSITORY_NAMelist	MQINQ
nazwa menedżera kolejek współużytkowanych			MQIA_SHARED_Q_Q_MGR_NAME,	MQINQ
Zdarzenie ssl			MQIA_SSL_EVENT,	MQINQ
fips ssl			MQIA_SSL_FIPS_REQUIRED	MQINQ
Licznik zerowania klucza SSL			MQIA_SSL_RESET_COUNT	MQINQ
początkowe zdarzenie zatrzymania			MQIA_START_STOP_EVENT	MQINQ
Interwał statystyki			MQIA_STATISTICS_INTERVAL	MQINQ
Dostępność punktu synchronizacji			MQIA_SYNCPOINT,	MQINQ

Atrybut	Struktura danych	Pole	Zapytanie	Wywołanie
kanały tcp			MQIA_TCP_CHANNELS	MQINQ
Podtrzymuj połączenie TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nazwa TCP			MQCA_TCP_NAME	MQINQ
Typ stosu TCP			MQIA_TCP_STACK_TYPE	MQINQ
Zapis śledzenia trasy			MQIA_TRACE_ROUTE_RECORDING	MQINQ
Interwał wyzwalacza			MQIA_TRIGGER_INTERVAL	MQINQ
ID użytkownika	Protokół MQCSP	CSPUserIdPtr		MQCONN
	Protokół MQCSP	Przesunięcie CSPUserId		MQCONN
	Protokół MQCSP	Długość CSPUserId		MQCONN

Odwołanie do nagłówka ImqReference

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord C++.

Atrybut	Struktura danych	Pole
środowisko docelowe	MQRMH	Długość DestEnv, Przesunięcie DestEnv
Nazwa miejsca docelowego	MQRMH	DestName(długość), Przesunięcie DestName
Identyfikator instancji	MQRMH	Identyfikator ObjectInstance
długość logiczna	MQRMH	Długość DataLogical
przesunięcie logiczne	MQRMH	Przesunięcie DataLogical
przesunięcie logiczne 2	MQRMH	DataLogicalOffset2
Typ odniesienia	MQRMH	ObjectType
Środowisko źródłowe	MQRMH	SrcEnvDługość, Przesunięcie SrcEnv
NAZWA ŹRÓDŁA	MQRMH	SrcNameDługość, Przesunięcie SrcName

ImqTrigger -referencja

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord C++.

Tabela 224. Odwołanie krzyżowe ImqTrigger		
Atrybut	Struktura danych	Pole
Identyfikator aplikacji	MQTM	ApplId
Typ aplikacji	MQTM	ApplType
Dane środowiska	MQTM	EnvData

Tabela 224. Odwołanie krzyżowe ImqTrigger (kontynuacja)		
Atrybut	Struktura danych	Pole
Nazwa procesu	MQTM	ProcessName
Nazwa kolejki	MQTM	Nazwa QName
Dane wyzwalacza	MQTM	TriggerData
Dane użytkownika	MQTM	UserData

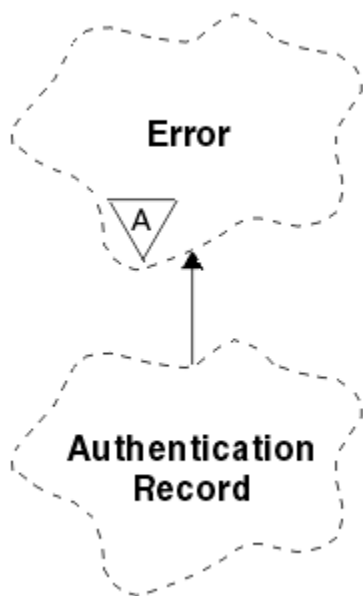
ImqWork-odwołanie do nagłówka

Odwołanie krzyżowe atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord C + +.

Atrybut	Struktura danych	Pole
znacznik komunikatu	MQWIH	MessageToken
nazwa usługi	MQWIH	ServiceName
krok usługi	MQWIH	ServiceStep

ImqAuthentication-rejestrowanie klasy C++

Ta klasa hermetykuje rekord informacji uwierzytelniających (MQAIR) do użycia podczas wykonywania metody ImqQueueManager: :connect, dla niestandardowych połączeń klienta SSL.



Rysunek 46. Klasa rekordu ImqAuthentication

Więcej szczegółowych informacji można znaleźć w opisie menedżera ImqQueue: :connect. Ta klasa nie jest dostępna na platformie z/OS .

- [“Atrybuty obiektu” na stronie 1939](#)
- [“Konstruktory” na stronie 1939](#)
- [“Metody obiektów \(publiczne\)” na stronie 1939](#)
- [“Metody obiektów \(chronione\)” na stronie 1940](#)

Atrybuty obiektu

Typ połączenia

Nazwa połączenia z serwerem CRL LDAP. Jest to adres IP lub nazwa DNS, po której opcjonalnie znajduje się numer portu, w nawiasach.

odwołanie do połączenia

Odwołanie do obiektu menedżera kolejek *ImqQueue*, który udostępnia wymagane połączenie z menedżerem kolejek (lokalnym). Wartością początkową jest zero. Nie należy mylić tej nazwy z nazwą menedżera kolejek, która identyfikuje menedżer kolejek (być może zdalny) dla określonej kolejki.

następny rekord uwierzytelniania

Następny obiekt tej klasy, w żadnym konkretnym zamówieniu, o tym samym **odwołaniu do połączenia**, co ten obiekt. Wartością początkową jest zero.

Hasło

Hasło podane do uwierzytelniania połączenia z serwerem CRL LDAP.

poprzedni rekord uwierzytelniania

Poprzedni obiekt tej klasy, w żadnym określonym porządku, o tym samym **odwołaniu do połączenia**, co ten obiekt. Wartością początkową jest zero.

Typ

Typ informacji uwierzytelniających zawartych w rekordzie.

nazwa użytkownika,

Identyfikator użytkownika podany do autoryzacji na serwerze CRL LDAP.

Konstruktory

ImqAuthenticationRecord ();

Konstruktor domyślny.

Metody obiektów (publiczne)

void operator = (const ImqAuthenticationRecord & *powietrze*);

Kopiuje dane instancji z *powietrza*, zastępując istniejące dane instancji.

const ImqString & connectionName () const;

Zwraca **nazwę połączenia**.

void setConnectionName (const ImqString & *nazwa*);

Ustawia **nazwę połączenia**.

void setConnectionName (const char * *nazwa* = 0);

Ustawia **nazwę połączenia**.

ImqQueueManager * connectionReference () const;

Zwraca **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager & *menedżer*);

Ustawia **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager * *menedżer* = 0);

Ustawia **odwołanie do połączenia**.

void copyOut (MQAIR * *pAir*);

Kopiuje dane instancji do produktu *pAir*, zastępując istniejące dane instancji. Może to wiązać się z przydzielaniem pamięci zależnej.

void clear (MQAIR * *pAir*);

Czyści strukturę i zwalnia pamięć zależną, do której odwołuje się produkt *pAir*.

Rekord ImqAuthenticationRecord * nextAuthenticationRecord () const;

Zwraca **następny rekord uwierzytelniania**.

const ImqString & password () const;

Zwraca **hasło**.

void setPassword (const ImqString & hasło);

Ustawia **hasło**.

void setPassword (const char * hasło = 0);

Ustawia **hasło**.

Rekord ImqAuthenticationRecord * previousAuthenticationRecord () const;

Zwraca **poprzedni rekord uwierzytelniania**.

Typ MQLONG () const;

Zwraca **typ**.

void setType (const MQLONG typ);

Ustawia **typ**.

const ImqString & userName () const;

Zwraca **nazwę użytkownika**.

void setUsername (const ImqString & nazwa);

Ustawia **nazwę użytkownika**.

void setUsername (const char * nazwa = 0);

Ustawia **nazwę użytkownika**.

Metody obiektów (chronione)

void setNextAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

Ustawia **następny rekord uwierzytelniania**.

Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik ma pewność, że nie będzie przerywać listy rekordów uwierzytelniania.

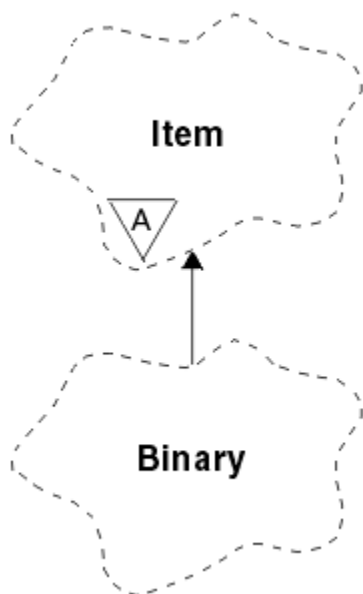
void setPreviousAuthenticationRecord (ImqAuthenticationRecord * pAir = 0);

Ustawia **poprzedni rekord uwierzytelniania**.

Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik ma pewność, że nie będzie przerywać listy rekordów uwierzytelniania.

Klasa ImqBinary C++

Ta klasa hermetykuje binarną tablicę bajtów, która może być używana dla wartości ImqMessage **token rozliczania, identyfikator korelacji identyfikator komunikatu** . Pozwala na łatwe przypisywanie, kopiowanie i porównywanie.



Rysunek 47. Klasa ImqBinary

- [“Atrybuty obiektu” na stronie 1941](#)
- [“Konstruktory” na stronie 1941](#)
- [“Przeciążone metody ImqItem” na stronie 1941](#)
- [“Metody obiektów \(publiczne\)” na stronie 1941](#)
- [“Metody obiektów \(chronione\)” na stronie 1942](#)
- [“Kody przyczyny” na stronie 1942](#)

Atrybuty obiektu

data

Tablica bajtów danych binarnych. Początkowa wartość jest równa null.

Długość danych

Liczba bajtów. Wartością początkową jest zero.

wskaźnik danych

Adres pierwszego bajtu **danych**. Wartością początkową jest zero.

Konstruktory

ImqBinary();

Konstruktor domyślny.

ImqBinary(const ImqBinary & binary);

Konstruktor kopiowania.

ImqBinary(const void * data, const size_t długość);

Kopiuje *długość* bajtów z *danych*.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & komunikat);

Kopiuje **dane** do buforu komunikatów, zastępując istniejącą treść. Ustawia wartość parametru *msg format* na wartość MQFMT_NONE.

Więcej szczegółów można znaleźć w opisie metody klasy ImqItem .

virtual ImqBoolean pasteIn (ImqMessage & komunikat);

Ustawia **dane** , przesyłając pozostałe dane z buforu komunikatów, zastępując istniejące **dane**.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_NONE.

Więcej szczegółów można znaleźć w opisie metody klasy ImqItem .

Metody obiektów (publiczne)

void operator = (const ImqBinary & binary);

Kopiuje bajty z *binary*.

ImqBoolean operator == (const ImqBinary & binary);

Porównuje ten obiekt z *binarnym*. Zwraca FALSE, jeśli nie jest równe i TRUE w przeciwnym razie.

Obiekty są równe, jeśli mają taką samą **długość danych** i są zgodne z liczbą bajtów.

ImqBoolean copyOut (void * buffer, const size_t length, const char pad = 0);

Kopiuje do *długość* bajtów ze **wskaźnika danych** do *buforu*. Jeśli **długość danych** jest niewystarczająca, pozostałą przestrzeń w polu *bufor* jest wypełniona *dopełnieniem* bajtów. Wartość *bufor* może wynosić zero, jeśli *długość* również wynosi zero. Wartość *długość* nie może być ujemna. Zwraca wartość PRAWDA, jeśli powiodła się.

size_t dataLength () const ;

Zwraca **długość danych**.

ImqBoolean setDataLength (const size_t *długość*);

Ustawia **długość danych**. Jeśli **długość danych** zostanie zmieniona w wyniku tej metody, dane w obiekcie są niezainicjowane. Zwraca wartość PRAWDA, jeśli powiodła się.

void * dataPointer () const ;

Zwraca **wskaźnik danych**.

ImqBoolean isNull () const ;

Zwraca wartość PRAWDA, jeśli **długość danych** wynosi zero lub jeśli wszystkie bajty **dane** są równe zero. W przeciwnym razie zwraca FALSE.

ImqBoolean set (const void * *buffer*, const size_t *długość*);

Kopiuje *długość* bajtów z *buforu*. Zwraca wartość PRAWDA, jeśli powiodła się.

Metody obiektów (chronione)**void clear ();**

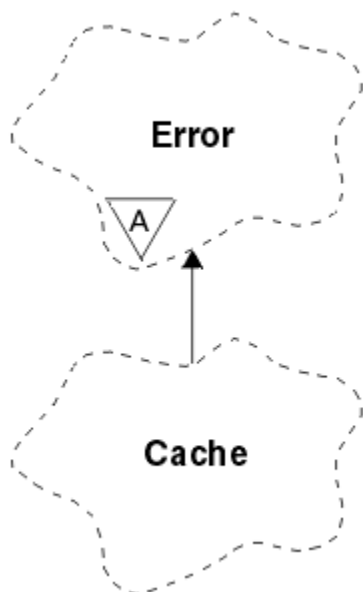
Redukuje **długość danych** do zera.

Kody przyczyny

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

Klasa ImqCache C++

Klasa ta służy do przechowywania lub zestawiania danych w pamięci.



Rysunek 48. Klasa ImqCache

Klasa ta służy do przechowywania lub zestawiania danych w pamięci. Możesz nominować bufor pamięci o stałej wielkości, albo system może automatycznie zapewnić elastyczną ilość pamięci. Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“ImqCache -referencja”](#) na stronie 1923.

- [“Atrybuty obiektu”](#) na stronie 1943
- [“Konstruktor”](#) na stronie 1943
- [“Metody obiektów \(publiczne\)”](#) na stronie 1943
- [“Kody przyczyny”](#) na stronie 1945

Atrybuty obiektu

bufor automatyczny

Wskazuje, czy pamięć buforu jest zarządzana automatycznie przez system (TRUE), czy też jest dostarczana przez użytkownika (FAŁSZ). Początkowo jest ona ustawiona na TRUE.

Ten atrybut nie jest ustawiony bezpośrednio. Jest on ustawiany pośrednio przy użyciu metody **useEmptyBuffer** lub **useFullBuffer**.

Jeśli podana jest pamięć masowa użytkownika, ten atrybut ma wartość FALSE, nie można powiększać pamięci buforu, a błędy przepełnienia buforu mogą wystąpić. Adres i długość buforu pozostają stałe.

Jeśli pamięć masowa użytkownika nie została podana, atrybut ten ma wartość TRUE, a pamięć buforu może zwiększać się przyrostowo w celu dostosowania do dowolnej ilości danych komunikatu. Jeśli jednak bufor rośnie, może to zmienić adres buforu, dlatego należy zachować ostrożność przy korzystaniu z **wskaźnika bufora** i **wskaźnika danych**.

długość buforu

Liczba bajtów pamięci w buforze. Wartością początkową jest zero.

wskaźnik buforu

Adres pamięci buforu. Początkowa wartość jest równa null.

Długość danych

Liczba bajtów, które powiodło się, **wskaźnik danych**. Wartość ta musi być równa lub mniejsza od **długości komunikatu**. Wartością początkową jest zero.

Pozycja danych

Liczba bajtów poprzedzających **wskaźnik danych**. Wartość ta musi być równa lub mniejsza od **długości komunikatu**. Wartością początkową jest zero.

wskaźnik danych

Adres części buforu, który ma zostać zapisany lub odczytany od następnego. Początkowa wartość jest równa null.

długość komunikatu

Liczba bajtów znaczących danych w buforze. Wartością początkową jest zero.

Konstruktory

ImqCache();

Konstruktor domyślny.

ImqCache(const ImqCache & *pamięć podręczna*);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqCache & *cache*);

Kopiuje do **długości komunikatu** bajtów danych z obiektu *pamięć podręczna* do obiektu. Jeśli parametr **bufor automatyczny** ma wartość FAŁSZ, **długość buforu** musi być już wystarczająca, aby pomieścić skopiowane dane.

ImqBoolean automaticBuffer () const ;

Zwraca wartość **automatic buffer** (automatyczny bufor).

size_t bufferSize () const ;

Zwraca **długość buforu**.

char * bufferPointer () const ;

Zwraca **wskaźnik buforu**.

void clearMessage ();

Ustawia **długość komunikatu** i **przesunięcie danych** na zero.

size_t dataLength () const ;

Zwraca **długość danych**.

size_t dataOffset () const ;

Zwraca **przesunięcie danych**.

ImqBoolean setDataOffset (const size_t przesunięcie);

Ustawia **przesunięcie danych**. **Długość komunikatu** jest zwiększana, jeśli jest to konieczne, aby zapewnić, że nie jest ona mniejsza niż **przesunięcie danych**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

char * dataPointer () const ;

Zwraca kopię **wskaźnika danych**.

size_t messageLength () const ;

Zwraca **długość komunikatu**.

ImqBoolean setMessageLength (const size_t długość);

Ustawia **długość komunikatu**. Zwiększa **długość buforu** , jeśli jest to konieczne dla zapewnienia, że **długość komunikatu** nie jest większa niż **długość buforu**. Redukuje **przesunięcie danych** , jeśli jest to konieczne, aby upewnić się, że nie jest ona większa niż **długość komunikatu**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean moreBytes (const size_t bajty-wymagane);

Zapewnia, że *bajty-wymagane* będą dostępne więcej bajtów (do zapisu) między **wskaźnikiem danych** a końcem buforu. Zwraca wartość PRAWDA, jeśli powiodła się.

Jeśli parametr **bufor automatyczny** ma wartość PRAWDA, więcej pamięci jest uzyskiwanych zgodnie z wymaganiami. W przeciwnym razie **długość buforu** musi być już odpowiednia.

ImqBoolean read (const size_t length, char * & external-buffer);

Kopiuje *długość* bajtów z buforu, począwszy od pozycji **wskaźnik danych** , do *zewnętrznego buforu*. Po skopiowaniu danych wartość **przesunięcie danych** jest zwiększana o *długość*. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean resizeBuffer (const size_t długość);

Zmienia **długość buforu** pod warunkiem, że **bufor automatyczny** ma wartość PRAWDA. Jest to osiągnięte przez realokację pamięci buforu. Do wartości **długość komunikatu** bajty danych z istniejącego buforu są kopiowane do nowego. Maksymalna liczba skopiowanych bajtów wynosi *długość* bajtów. **Wskaźnik buforu** jest zmieniany. **Długość komunikatu** i **przesunięcie danych** są zachowywane tak blisko, jak to jest możliwe w granicach nowego buforu. Zwraca TRUE, jeśli powiedzie się, a FALSE, jeśli **bufor automatyczny** ma wartość FALSE.

Uwaga: Ta metoda może nie powieść się z MQRC_STORAGE_NOT_AVAILABLE, jeśli występuje problem z zasobami systemowymi.

ImqBoolean useEmptyBuffer (const char * external-buffer, const size_t długość);

Identyfikuje pusty bufor użytkownika, ustawiając **wskaźnik buforu** tak, aby wskazywał na *external-buffer*, **długość buforu** na *długość*, a **długość komunikatu** na zero. Wykonuje komendę **clearMessage**. Jeśli bufor jest w pełni załadowany danymi, zamiast tego należy użyć metody **useFullBuffer** . Jeśli bufor jest częściowo załadowany danymi, należy użyć metody **setMessageLength** , aby wskazać poprawną kwotę. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Tej metody można użyć do określenia stałej wielkości pamięci, tak jak opisano to wcześniej (*external-buffer* nie ma wartości null, a *length* jest niezerową), w takim przypadku **automatic buffer** ma wartość FALSE lub może zostać użyty do przywrócenia elastycznej pamięci zarządzanej przez system (*external-buffer* ma wartość NULL, a *długość* wynosi zero), w którym to przypadku **bufor automatyczny** jest ustawiony na wartość TRUE.

ImqBoolean useFullBuffer (const char * externalBuffer, const size_t długość);

Tak jak w przypadku produktu **useEmptyBufor**, z tą różnicą, że **długość komunikatu** jest ustawiona na wartość *długość*. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean write (const size_t długość, const char * external-buffer);

Kopiuje *długość* bajtów z *zewnętrznego-buforu* do buforu rozpoczynając od pozycji **wskaźnik danych** . Po skopiowaniu danych **przesunięcie danych** jest zwiększane o *długość*, a **długość komunikatu** jest zwiększana, jeśli jest to konieczne, aby zapewnić, że nie jest ona mniejsza niż nowa wartość **przesunięcia danych** . Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

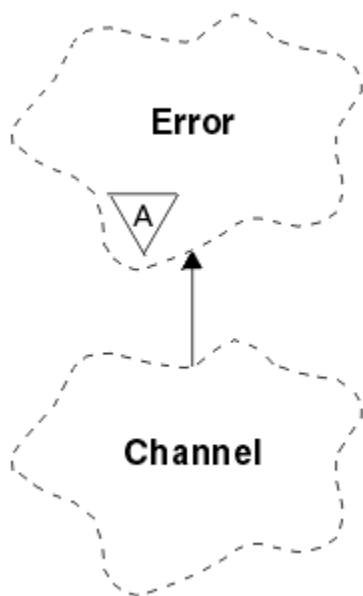
Jeśli **bufor automatyczny** ma wartość PRAWDA, gwarantowana jest odpowiednia ilość pamięci. W przeciwnym razie, ostateczne **przesunięcie danych** nie może przekraczać **długości buforu**.

Kody przyczyny

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_OBCIĘTY
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_ZERO_LENGTH

Klasa języka C++ ImqChannel

Ta klasa hermetyzuje definicję kanału (MQCD) do użycia podczas wykonywania menedżera: :connect, dla niestandardowych połączeń klientów.



Rysunek 49. Klasa ImqChannel

Więcej szczegółów można znaleźć w opisie menedżera: :connect, a także [Przykładowy program HELLO WORLD \(imqwrld.cpp\)](#).

Nie wszystkie wymienione metody mają zastosowanie do wszystkich platform. Aby uzyskać więcej informacji, zapoznaj się z opisami komend [DEFINE CHANNEL](#) i [ALTER CHANNEL](#).

Klasa ImqChannel nie jest obsługiwana w systemie z/OS.

- [“Atrybuty obiektu” na stronie 1946](#)
- [“Konstruktor” na stronie 1947](#)
- [“Metody obiektów \(publiczne\)” na stronie 1947](#)
- [“Kody przyczyny” na stronie 1950](#)

Atrybuty obiektu

wsadowe bicie serca

Liczba milisekund między operacjami sprawdzania, czy kanał zdalny jest aktywny. Wartością początkową jest 0.

nazwa kanału

Nazwa kanału. Początkowa wartość jest równa null.

Typ połączenia

Nazwa połączenia. Na przykład adres IP komputera hosta. Początkowa wartość jest równa null.

Kompresja nagłówka

Lista technik kompresji danych nagłówka obsługiwanych przez kanał. Wartości początkowe są ustawione na wartość MQCOMPRESS_NOT_AVAILABLE.

interwał pulsu

Liczba sekund między operacjami sprawdzania, czy połączenie nadal działa. Wartością początkową jest 300.

Interwał sprawdzania połączenia

Liczba sekund przekazywana do stosu komunikacyjnego określająca czas utrzymywania połączenia dla kanału. Wartością początkową jest MQKAI_AUTO.

Adres lokalny

Adres komunikacji lokalnej dla kanału.

Maksymalna długość komunikatu

Maksymalna długość komunikatu obsługiwana przez kanał w pojedynczej komunikacji. Wartość początkowa to 4 194 304.

Kompresja komunikatu

Lista technik kompresji danych komunikatu obsługiwanych przez kanał. Wartości początkowe są ustawione na wartość MQCOMPRESS_NOT_AVAILABLE.

Nazwa trybu

Nazwa trybu. Początkowa wartość jest równa null.

Hasło

Hasło podane na potrzeby uwierzytelniania połączenia. Początkowa wartość jest równa null.

liczba operacji wyjścia odbierania

Liczba wyjść odbierania. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

odbieranie nazw wyjść

Nazwy wyjść odbierania.

odbieranie danych użytkownika

Dane powiązane z wyjściami odbioru.

Nazwa wyjścia zabezpieczeń

Nazwa wyjścia zabezpieczeń, które ma zostać wywołane po stronie serwera połączenia. Początkowa wartość jest równa null.

dane użytkownika ochrony

Dane, które mają zostać przekazane do wyjścia zabezpieczeń. Początkowa wartość jest równa null.

liczba operacji wyjścia wysyłania

Liczba wyjść wysyłania. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

wysyłanie nazw wyjść

Nazwy wyjść nadawanych.

wysyłanie danych użytkownika

Dane powiązane z wyjściami nadawczym.

CipherSpec SSL

Atrybut CipherSpec do użycia z protokołem SSL.

Typ uwierzytelniania klienta SSL

Typ uwierzytelniania klienta używany z protokołem SSL.

Nazwa węzła sieci SSL

Nazwa węzła sieci używana z protokołem SSL.

Nazwa programu transakcyjnego

Nazwa programu transakcyjnego. Początkowa wartość jest równa null.

Typ transportu

Typ transportu połączenia. Początkowa wartość to MQXPT_LU62.

ID użytkownika

Identyfikator użytkownika podany do autoryzacji. Początkowa wartość jest równa null.

Konstruktory

ImqChannel() ;

Konstruktor domyślny.

ImqChannel(const ImqChannel & kanał);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqChannel & kanał);

Kopiuje dane instancji z *kanału*, zastępując wszystkie istniejące dane instancji.

MQLONG batchHeartBeat () const;

Zwraca **wsadowe bicie serca**.

ImqBoolean setBatchHeartBeat(const MQLONG puls = 0L);

Ustawia **batch heart-beat**(wsadowe bicie serca). Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString channelName() const;

Zwraca **nazwę kanału**.

ImqBoolean setChannelNazwa (const char * nazwa = 0);

Ustawia **nazwę kanału**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString connectionName() const;

Zwraca **nazwę połączenia**.

ImqBoolean setConnectionNazwa (const char * nazwa = 0);

Ustawia **nazwę połączenia**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

size_t headerCompressionCount () const;

Zwraca liczbę obsługiwanych technik kompresji danych nagłówka.

ImqBoolean headerCompression(const size_t count, MQLONG compress []) const;

Zwraca kopie obsługiwanych technik kompresji danych nagłówka w pliku **compress**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setHeaderKompresja (const size_t count, const MQLONG compress []);

Ustawia obsługiwane techniki kompresji danych nagłówka na **compress**.

Ustawia liczbę obsługiwanych technik kompresji danych nagłówka na **count**.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG heartBeatInterwał () const;

Zwraca wartość **interwał pulsu**.

ImqBoolean setHeartBeatInterval(const MQLONG interwał = 300L);

Ustawia **interwał pulsu**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG keepAliveInterwał () const;

Zwraca wartość **interwał sprawdzania połączenia**.

ImqBoolean setKeepAliveInterval(const MQLONG interwał = MQKAI_AUTO);

Ustawia **interwał sprawdzania połączenia**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString localAddress() const;

Zwraca **adres lokalny**.

ImqBoolean setLocalAddress (const char * adres = 0);

Ustawia **adres lokalny**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumMessageLength () const;

Zwraca **maksymalną długość komunikatu**.

ImqBoolean setMaximumMessageLength(const MQLONG długość = 4194304L);

Ustawia **maksymalną długość komunikatu**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

size_t messageCompressionCount () const;

Zwraca liczbę obsługiwanych technik kompresji danych komunikatu.

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const;

Zwraca kopie obsługiwanych technik kompresji danych komunikatu w pliku **compress**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setMessageKompresja (const size_t count, const MQLONG compress []);

Służy do ustawiania obsługiwanych technik kompresji danych komunikatu w celu kompresji.

Ustawia liczbę obsługiwanych technik kompresji danych komunikatu do zliczania.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString modeName() const;

Zwraca **nazwę trybu**.

ImqBoolean setModeNazwa (const char * nazwa = 0);

Ustawia **nazwę trybu**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString hasło () const;

Zwraca **hasło**.

ImqBoolean setPassword(const char * hasło = 0);

Ustawia **hasło**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

size_t receiveExitCount () const;

Zwraca **liczbę wyjść odbierania**.

ImqString receiveExitNazwa ();

Zwraca pierwszą z **nazw wyjścia odbierania** (jeśli istnieje). Jeśli **liczba operacji wyjścia odbierania** wynosi zero, zwraca pusty łańcuch.

ImqBoolean receiveExitNames (const size_t liczba, ImqString * nazwy_nazw []);

Zwraca kopie **nazw wyjścia odbierania** w **nazwy_nazw**. Ustawia dowolną wartość **names** przekraczając **receive exit count** na łańcuchy o wartości NULL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveExitName(const char * nazwa = 0);

Ustawia **nazwy wyjścia odbierania** na pojedynczą **nazwę**. Wartość **nazwa** może być pusta lub mieć wartość NULL. Ustawia wartość parametru **receive exit count** na wartość 1 lub zero. Kasuje **odbieranie danych użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveExitNames(const size_t liczba, const char * nazwy_nazw []);

Ustawia **nazwy wyjścia odbierania** na **nazwy**. Poszczególne wartości **nazwy** nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę operacji wyjścia odbierania** na wartość **liczba**. Kasuje **odbieranie danych użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveExitNames(const size_t liczba, const ImqString * nazwy_nazw []);

Ustawia **nazwy wyjścia odbierania** na **nazwy**. Poszczególne wartości **nazwy** nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę operacji wyjścia odbierania** na wartość **liczba**. Kasuje **odbieranie danych użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString receiveUserData ();

Zwraca pierwszą z pozycji **receive user data** (odbieranie danych użytkownika), jeśli istnieje. Jeśli **liczba operacji wyjścia odbierania** wynosi zero, zwraca pusty łańcuch.

ImqBoolean receiveUserData (const size_t *liczba*, ImqString * *dane* []);

Zwraca kopie elementów **receive user data** (odbieranie danych użytkownika) w *danych*. Ustawia *dane* przekraczające **liczbę wyjść odbierania** do łańcuchów o wartości NULL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveUserData(const char * *data* = 0);

Ustawia **dane użytkownika odbieranego** na pojedynczy element *dane*. Jeśli parametr *data* ma wartość inną niż NULL, **liczba operacji wyjścia odbierania** musi być równa co najmniej 1. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveUserData(const size_t *liczba*, const char * *data* []);

Ustawia **dane użytkownika odbieranego** na *dane*. Wartość *liczba* nie może być większa niż wartość **liczba wyjść odbierania**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setReceiveUserData(const size_t *count*, const ImqString * *data* []);

Ustawia **dane użytkownika odbieranego** na *dane*. Wartość *liczba* nie może być większa niż wartość **liczba wyjść odbierania**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString securityExitNazwa () const;

Zwraca **nazwę wyjścia zabezpieczeń**.

ImqBoolean setSecurityExitName(const char * *nazwa* = 0);

Ustawia **nazwę wyjścia zabezpieczeń**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString securityUserData () const;

Zwraca **dane użytkownika zabezpieczeń**.

ImqBoolean setSecurityUserData(const char * *data* = 0);

Ustawia **dane użytkownika zabezpieczeń**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

size_t sendExitCount () const;

Zwraca **liczbę wyjść wysyłania**.

ImqString sendExitNazwa ();

Zwraca pierwszą z opcji **nazwy wyjścia wysyłania** (jeśli istnieje). Zwraca pusty łańcuch, jeśli **liczba operacji wyjścia wysyłania** wynosi zero.

ImqBoolean sendExitNazwy (const size_t *liczba*, ImqString * *nazwy_nazw* []);

Zwraca kopie **nazw wyjścia wysyłania** w *nazwy_nazw*. Ustawia dowolną wartość *names* przekraczając **send exit count** (liczba wyjść wysyłania) do łańcuchów o wartości NULL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setSendExitName(const char * *nazwa* = 0);

Ustawia wartość parametru **send exit names** na pojedynczą *name*. Wartość *nazwa* może być pusta lub mieć wartość NULL. Ustawia wartość parametru **send exit count** na wartość 1 lub zero. Czyści **wysyłanie danych użytkownika**. Ta metoda zwraca TRUE w przypadku powodzenia

ImqBoolean setSendExitNames(const size_t *liczba*, const char * *nazwy_nazw* []);

Ustawia **nazwy wyjścia wysyłania** na *nazwy*. Poszczególne wartości *nazwy* nie mogą być puste ani mieć wartości NULL. Ustawia wartość parametru **send exit count** na *licznik*. Czyści **wysyłanie danych użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setSendExitNames(const size_t *count*, const ImqString * *names* []);

Ustawia **nazwy wyjścia wysyłania** na *nazwy*. Poszczególne wartości *nazwy* nie mogą być puste ani mieć wartości NULL. Ustawia wartość parametru **send exit count** na *licznik*. Czyści **wysyłanie danych użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString sendUserData ();

Zwraca pierwszą z pozycji **send user data** (wysyłanie danych użytkownika), jeśli istnieje. Zwraca pusty łańcuch, jeśli **liczba operacji wyjścia wysyłania** wynosi zero.

ImqBoolean sendUserData (const size_t *licznik*, ImqString * *dane* []);

Zwraca kopie elementów **wysyłaj dane użytkownika** w *danych*. Ustawia *dane* przekraczające wartość **send exit count** (liczba wyjść wysyłania) do łańcuchów o wartości NULL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setSendUserData(const char * *data* = 0);

Ustawia **wysyłanie danych użytkownika** na pojedynczy element *dane*. Jeśli parametr *data* ma wartość inną niż NULL, wartość **liczba operacji wyjścia wysyłania** musi wynosić co najmniej 1. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setSendUserData(const size_t *liczba*, const char * *data* []);

Ustawia **wysyłanie danych użytkownika** na *dane*. Wartość *liczba* nie może być większa niż wartość **liczba wyjść wysyłania**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setSendUserData(const size_t *count*, const ImqString * *data* []);

Ustawia **wysyłanie danych użytkownika** na *dane*. Wartość *liczba* nie może być większa niż wartość **liczba wyjść wysyłania**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString sslCipherSpecyfikacja () const;

Zwraca specyfikację szyfru SSL.

ImqBoolean setSslCipherSpecification(const char * *nazwa* = 0);

Ustawia specyfikację szyfru SSL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG sslClientAuthentication () const;

Zwraca typ uwierzytelniania klienta SSL.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);

Ustawia typ uwierzytelniania klienta SSL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString sslPeerNazwa () const;

Zwraca nazwę węzła sieci SSL.

ImqBoolean setSslPeerName(const char * *nazwa* = 0);

Ustawia nazwę węzła sieci SSL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString transactionProgramName () const;

Zwraca **nazwę programu transakcyjnego**.

ImqBoolean setTransactionProgramName(const char * *nazwa* = 0);

Ustawia **nazwę programu transakcyjnego**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG transportType() const;

Zwraca **typ transportu**.

ImqBoolean setTransportType (const MQLONG *typ* = MQXPT_LU62);

Ustawia **typ transportu**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString userId() const;

Zwraca **ID użytkownika**.

ImqBoolean setUserId (const char * *id* = 0);

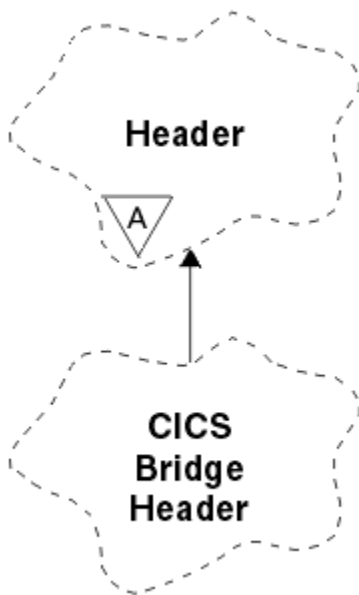
Ustawia **ID użytkownika**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Kody przyczyny

- Błąd MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridgeHeader C++

Ta klasa hermetyzuje określone funkcje struktury danych MQCIH.



Rysunek 50. Klasa *ImqCICSBridgeHeader*

Obiekty tej klasy są używane przez aplikacje, które wysyłają komunikaty do CICS bridge za pośrednictwem IBM MQ for z/OS.

- [“Atrybuty obiektu” na stronie 1951](#)
- [“Konstruktor” na stronie 1954](#)
- [“Przeciążone metody ImqItem” na stronie 1954](#)
- [“Metody obiektów \(publiczne\)” na stronie 1954](#)
- [“Dane obiektu \(chronione\)” na stronie 1956](#)
- [“Kody przyczyny” na stronie 1956](#)
- [“Kody powrotu” na stronie 1957](#)

Atrybuty obiektu

Deskryptor ADS

Wysyłanie/odbieranie deskryptora ADS. Ten parametr jest ustawiany za pomocą komendy MQCADSD_NONE. Wartością początkową jest MQCADSD_NONE. Możliwe są następujące wartości dodatkowe:

- MQCADSD_NONE
- MQCADSD_SEND,
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

identyfikator uwagi

Klawisz AID. Pole musi mieć długość MQ_ATTENTION_ID_LENGTH.

element uwierzytelniający

RACF (hasło lub passticket). Wartość początkowa zawiera odstępy, o długości MQ_AUTHENTICATOR_LENGTH.

kod abend mostu

Kod abend mostu, o długości MQ_ABEND_CODE_LENGTH. Wartość początkowa to cztery puste znaki. Wartość zwracana w tym polu jest zależna od kodu powrotu. Więcej informacji na temat zawiera sekcja [Tabela 225 na stronie 1957](#).

kod anulowania mostu

Kod transakcji abend mostu. Pole jest zastrzeżone, musi zawierać spacje i musi mieć długość MQ_CANCEL_CODE_LENGTH.

kod zakończenia mostu

Kod zakończenia, który może zawierać kod zakończenia IBM MQ lub wartość CICS EIBRESP. Pole ma początkową wartość MQCC_OK. Wartość zwracana w tym polu jest zależna od kodu powrotu. Więcej informacji na temat zawiera sekcja [Tabela 225 na stronie 1957](#).

przesunięcie błędu mostu

Przesunięcie błędu mostu. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

kod przyczyny mostu

Kod przyczyny. To pole może zawierać przyczynę IBM MQ lub wartość CICS EIBRESP2 . Pole ma początkową wartość parametru MQRC_NONE. Wartość zwracana w tym polu jest zależna od kodu powrotu. Więcej informacji na temat zawiera sekcja [Tabela 225 na stronie 1957](#).

kod powrotu mostu

Kod powrotu z CICS bridge. Wartością początkową jest MQCRC_OK.

zadanie konwersacyjne

Określa, czy zadanie może być konwersacyjne. Wartością początkową jest MQCCT_NO. Możliwe są następujące wartości dodatkowe:

- MQCCT_YES
- MQCCT_NO

pozycja kursora

Pozycja kursora. Wartością początkową jest zero.

czas przechowywania obiektu

CICS bridge czas wydania instalacji.

narzędzie, takie jak

Atrybut emulowany terminalu. Pole musi mieć długość MQ_FACILITY_LIKE_LENGTH.

obiekt token

Wartość znacznika BVT. Pole musi mieć długość MQ_FACILITY_LENGTH. Wartością początkową jest MQCFAC_NONE.

funkcja

Funkcja, która może zawierać nazwę wywołania IBM MQ lub funkcję CICS EIBFN. Pole ma początkową wartość MQCFUNC_NONE, o długości MQ_FUNCTION_LENGTH. Wartość zwracana w tym polu jest zależna od kodu powrotu. Więcej informacji na temat zawiera sekcja [Tabela 225 na stronie 1957](#).

Następujące dodatkowe wartości są możliwe, gdy **funkcja** zawiera nazwę wywołania IBM MQ :

- MQCFUNC_MQCONN
- MQCFUNC_MQGET-MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

okres oczekiwania na pobranie

Przedział czasu oczekiwania na wywołanie MQGET wystawione przez zadanie CICS bridge . Wartością początkową jest MQCGWI_DEFAULT. To pole ma zastosowanie tylko wtedy, gdy parametr **uow control** ma wartość MQCUOWC_FIRST. Możliwe są następujące wartości dodatkowe:

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

Typ odsyłacza

Typ odsyłacza. Wartością początkową jest MQCLT_PROGRAM. Możliwe są następujące wartości dodatkowe:

- PROGRAM MQCLT_PROGRAM
- MQCLT_TRANSACTION,

następny identyfikator transakcji

Identyfikator następnej transakcji do dołączenia. Pole musi mieć długość MQ_TRANSACTION_ID_LENGTH.

długość danych wyjściowych

Długość danych COMMAREA. Wartością początkową jest MQCODL_AS_INPUT.

format odpowiedzi

Nazwa formatu komunikatu odpowiedzi. Wartością początkową jest MQFMT_NONE o długości MQ_FORMAT_LENGTH.

kod początkowy

Kod początkowy transakcji. Pole musi mieć długość MQ_START_CODE_LENGTH. Wartością początkową jest MQCSC_NONE. Możliwe są następujące wartości dodatkowe:

- MQCSC_START
- MQCSC_STARTDATA,
- MQCSC_TERMINPUT
- MQCSC_NONE

status zakończenia zadania

Status zakończenia zadania. Wartością początkową jest MQCTES_NOSYNC. Możliwe są następujące wartości dodatkowe:

- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

Identyfikator transakcji

Identyfikator transakcji, która ma zostać przyłączona. Wartość początkowa musi zawierać odstępy, a wartość musi mieć długość MQ_TRANSACTION_ID_LENGTH. To pole ma zastosowanie tylko wtedy, gdy właściwość **uow control** ma wartość MQCUOWC_FIRST lub MQCUOWC_ONLY.

Sterowanie UOW

Element sterujący UOW. Wartością początkową jest MQCUOWC_ONLY. Możliwe są następujące wartości dodatkowe:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

wersja

Numer wersji MQCIH. Początkowa wartość to MQCIH_VERSION_2. Jediną inną obsługiwaną wartością jest MQCIH_VERSION_1.

Konstruktory

ImqCICSBridgeHeader();

Konstruktor domyślny.

ImqCICSBridgeHeader(const ImqCICSBridgeHeader & header);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut(ImqMessage & komunikat);

Wstawia strukturę danych MQCIH do buforu komunikatów na początku, dalej przesuwając istniejące dane komunikatu i ustawia format komunikatu na wartość MQFMT_CICS.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

virtual ImqBoolean pasteIn(ImqMessage & komunikat);

Odczytuje strukturę danych MQCIH z bufora komunikatów. Aby możliwe było pomyślne, kodowanie obiektu *msg* musi mieć wartość MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT na wartość MQENC_NATIVE. Aby możliwe było pomyślne działanie, format ImqMessage musi być następujący: MQFMT_CICS.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

Metody obiektów (publiczne)

void operator = (const ImqCICSBridgeHeader & header);

Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

MQLONG-Deskryptor ADSDescriptor () const;

Zwraca kopię deskryptora ADS.

void setADSDescriptor(const MQLONG deskryptor = MQCADSD_NONE);

Ustawia deskryptor ADS.

ImqString attentionIdentifier() const;

Zwraca kopię identyfikatora uwagi, dopełnianą spacjami kończącymi na długości MQ_ATTENTION_ID_LENGTH.

void setAttentionIdentifier (const char * dane = 0);

Ustawia identyfikator uwagi, dopełniony odstępami końcowymi na długość MQ_ATTENTION_ID_LENGTH. Jeśli nie zostanie podany *dane*, zresetuj identyfikator uwagi do wartości początkowej.

ImqString element uwierzytelniający () const;

Zwraca kopię elementu element uwierzytelniający dopełnione odstępami końcowymi na długości MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * dane = 0);

Ustawia parametr authenticator, dopełniony odstępami końcowymi na długość MQ_AUTHENTICATOR_LENGTH. Jeśli nie zostanie podany *dane*, resetuje element uwierzytelniający do wartości początkowej.

ImqString bridgeAbendCode () const;

Zwraca kopię kodu abend mostu dopełnione odstępami końcowymi na długości MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelCode () const;

Zwraca kopię kodu anulowania mostu dopełnione odstępami końcowymi na długości MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char * data = 0);

Ustawia kod anulowania mostu, dopełniony odstępami końcowymi na długość MQ_CANCEL_CODE_LENGTH. Jeśli nie podano *danych*, resetuje kod anulowania mostu do wartości początkowej.

MQLONG bridgeCompletion() const;

Zwraca kopię kodu zakończenia mostu.

MQLONG bridgeErrorPrzesunięcie () const;

Zwraca kopię przesunięcia błędu mostu.

MQLONG bridgeReasonCode () const;

Zwraca kopię kodu przyczyny mostu.

MQLONG bridgeReturn() const;

Zwraca kod powrotu mostu.

MQLONG conversationalTask() const;

Zwraca kopię zadania konwersacyjnego.

void setConversationalTask (const MQLONG zadanie = MQCCT_NO);

Ustawia zadanie konwersacyjne.

MQLONG cursorPosition() const;

Zwraca kopię pozycji kursora.

void setCursorPosition (const MQLONG pozycja = 0);

Ustawia pozycję kursora.

MQLONG facilityKeep() const;

Zwraca kopię czasu przechowywania obiektu.

void setFacilityKeepTime(const MQLONG czas = 0);

Ustawia czas przechowywania narzędzia.

ImqString facilityLike() const;

Zwraca kopię narzędzia, na przykład, dopełnione odstępami końcowymi do długości MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike (const char * nazwa = 0);

Ustawia narzędzie, takie jak, dopełniane odstępami końcowymi na długość MQ_FACILITY_LIKE_LENGTH. Jeśli nazwa nie zostanie podana, resetuje obiekt podobny do wartości początkowej.

ImqBinary facilityToken() const;

Zwraca kopię znacznika narzędzia.

ImqBoolean setFacilityToken (const ImqBinary & token);

Ustawia znacznik narzędzia. Parametr długość danych elementu token musi mieć wartość zero lub MQ_FACILITY_LENGTH. Zwraca wartość PRAWDA, jeśli powiodła się.

void setFacilityToken (const MQBYTE8 token = 0);

Ustawia znacznik narzędzia. token może być zerem, który jest taki sam, jak parametr MQCFAC_NONE. Jeśli element token ma wartość niezerową, musi on być adresowany do bajtów MQ_FACILITY_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQCFAC_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu. Na przykład (MQBYTE *) MQCFAC_NONE.

Funkcja ImqString () const;

Zwraca kopię funkcji dopełnianą odstępami końcowymi do długości MQ_FUNCTION_LENGTH.

Odstęp czasu getWaitMQLONG () const;

Zwraca kopię przedziału czasu oczekiwania na pobranie.

void setGetWaitInterval(const MQLONG interwał = MQCGWI_DEFA

Ustawia interwał oczekiwania na pobranie.

MQLONG linkType() const;

Zwraca kopię typu odsyłacza.

void setLinkType (const MQLONG typ = MQCLT_PROGRAM);

Ustawia typ odsyłacza.

ImqString nextTransactionIdentyfikator () const;

Zwraca kopię danych następnego identyfikatora transakcji, dopełnionych odstępami końcowymi na długość MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLength () const;

Zwraca kopię **długości danych wyjściowych**.

void setOutputDataLength(const MQLONG *długość* = MQCODL_AS_INPUT);

Ustawia **długość danych wyjściowych**.

ImqString replyToFormat () const;

Zwraca kopię nazwy **format odpowiedzi** , dopełnianą odstępami końcowymi do długości MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nazwa* = 0);

Ustawia wartość parametru **reply-to format**, uzupełniając odstępami końcowymi na długość MQ_FORMAT_LENGTH. Jeśli nie zostanie podana nazwa *nazwa* , resetuje **format odpowiedzi** do wartości początkowej.

ImqString startCode() const;

Zwraca kopię **kodu początkowego** dopełnianą odstępami końcowymi do długości MQ_START_CODE_LENGTH.

void setStartCode (const char * *dane* = 0);

Ustawia dane **kodu początkowego** , dopełniane odstępami końcowymi na długość MQ_START_CODE_LENGTH. Jeśli nie zostanie podany *dane* , zresetuj **kod początkowy** do wartości początkowej.

MQLONG taskEndStatus () const;

Zwraca kopię **statusu zakończenia zadania**.

ImqString transactionIdentifier() const;

Zwraca kopię danych **identyfikatora transakcji** dopełnionych spacjąmi kończącymi na długości MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentyfikator (const char * *dane* = 0);

Ustawia **identyfikator transakcji**, dopełniony odstępami końcowymi na długości MQ_TRANSACTION_ID_LENGTH. Jeśli nie zostanie podany *dane* , zresetuj **identyfikator transakcji** do wartości początkowej.

MQLONG UOWControl () const;

Zwraca kopię **elementu sterującego UOW**.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);

Ustawia element **Sterowanie UOW**.

Wersja MQLONG () const;

Zwraca numer **wersji** .

ImqBoolean setVersion(const MQLONG *wersja* = MQCIH_VERSION_2);

Ustawia numer **version** . Zwraca wartość PRAWDA, jeśli powiodła się.

Dane obiektu (chronione)

MQLONG olVersion

Maksymalny numer wersji MQCIH, który może być zakwaterowany w pamięci masowej przydzielonej dla *opcih*.

PMQCIH opcih

Adres struktury danych MQCIH. Ilość przydzielonej pamięci masowej jest wskazywana przez program *olVersion*.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_WRONG_VERSION

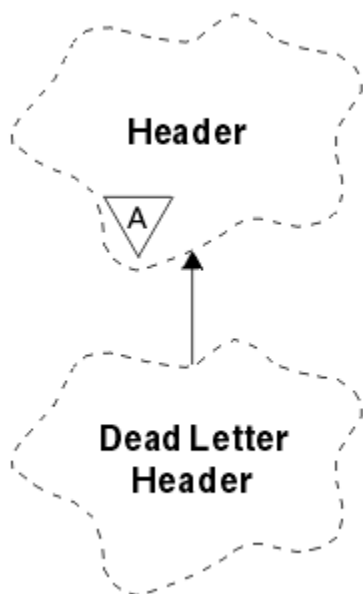
Kody powrotu

Tabela 225. Kody powrotu klasy ImqCICSBridgeHeader

Kod powrotu	Funkcja	CompCode	Przyczyna	Kod abend
MQCRC_OK				
MQCRC_BRIDGE_ERROR			MQFB_CICS	
BŁĄD MQCRC_MQ_API_ERROR	Nazwa wywołania IBM MQ	IBM MQ CompCode	IBM MQ Przyczyna	
MQCRC_BRIDGE_TIMEOUT	Nazwa wywołania IBM MQ	IBM MQ CompCode	IBM MQ Przyczyna	
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

Klasa języka C++ ImqDeadLetterHeader

Ta klasa hermetyzuje cechy struktury danych MQDLH.



Rysunek 51. Klasa ImqDeadLetterHeader

Obiekty tej klasy są zwykle używane przez aplikację, która napotka komunikat, który nie może zostać przetworzony. Nowy komunikat składający się z nagłówka niedostarczonych komunikatów i treści komunikatu jest umieszczany w kolejce niedostarczonych komunikatów, a komunikat jest odrzucany.

- [“Atrybuty obiektu” na stronie 1958](#)

- [“Konstruktory” na stronie 1958](#)
- [“Przeciążone metody ImqItem” na stronie 1958](#)
- [“Metody obiektów \(publiczne\)” na stronie 1959](#)
- [“Dane obiektu \(chronione\)” na stronie 1959](#)
- [“Kody przyczyny” na stronie 1959](#)

Atrybuty obiektu

dead-letter Kod przyczyny

Przyczyna, dla której komunikat dotarł do kolejki niedostarczonych komunikatów. Wartością początkową jest MQRC_NONE.

Nazwa menedżera kolejek docelowych

Nazwa oryginalnego docelowego menedżera kolejek. Nazwa to łańcuch o długości MQ_Q_MGR_NAME_LENGTH. Jej początkowa wartość jest równa null.

nazwa kolejki docelowej

Nazwa oryginalnej kolejki docelowej. Nazwa to łańcuch o długości MQ_Q_NAME_LENGTH. Jej początkowa wartość jest równa null.

Nazwa aplikacji wstawiającej

Nazwa aplikacji, która wstawiła komunikat do kolejki niedostarczonych komunikatów. Nazwa to łańcuch o długości MQ_PUT_APPL_NAME_LENGTH. Jej początkowa wartość jest równa null.

Typ aplikacji wstawiającej

Typ aplikacji, która umieszczała komunikat w kolejce niedostarczonych komunikatów. Wartością początkową jest zero.

Data wstawienia

Data wstawienia komunikatu do kolejki niedostarczonych komunikatów. Data to łańcuch o długości MQ_PUT_DATE_LENGTH. Jej początkowa wartość jest łańcuchem pustym.

Czas wstawienia

Czas wstawienia komunikatu do kolejki niedostarczonych komunikatów. Czas to łańcuch o długości MQ_PUT_TIME_LENGTH. Jej początkowa wartość jest łańcuchem pustym.

Konstruktory

ImqDeadLetterHeader();

Konstruktor domyślny.

ImqDeadLetterHeader(const ImqDeadLetterHeader & header);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & komunikat);

Wstawia strukturę danych MQDLH do buforu komunikatów na początku, a następnie dalej przenosi istniejące dane komunikatu. Ustawia wartość parametru **msg format** na wartość MQFMT_DEAD_LETTER_HEADER.

Więcej szczegółowych informacji można znaleźć w opisie metody klasy ImqHeader na stronie [“Klasa języka C++ ImqHeader” na stronie 1965](#).

virtual ImqBoolean pasteIn (ImqMessage & komunikat);

Odczytuje strukturę danych MQDLH z buforu komunikatów.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_DEAD_LETTER_HEADER.

Więcej szczegółowych informacji można znaleźć w opisie metody klasy ImqHeader na stronie [“Klasa języka C++ ImqHeader” na stronie 1965](#).

Metody obiektów (publiczne)

void operator = (const ImqDeadLetterHeader & header);

Kopiowanie danych instancji jest kopiowane z *nagłówka*, zastępując istniejące dane instancji.

MQLONG deadLetterReasonCode () const ;

Zwraca **kod przyczyny niedostarczonych komunikatów**.

void setDeadLetterReasonCode (const MQLONG przyczyna);

Ustawia **dead-letter reason code**.

ImqString destinationQueueManagerName () const ;

Zwraca **nazwę menedżera kolejek docelowych**, która jest pozbawiona końcowych odstępów.

void setDestinationQueueManagerName (const char * nazwa);

Ustawia **nazwę docelowego menedżera kolejek**. Obcinanie danych jest dłuższe niż wartość MQ_Q_MGR_NAME_LENGTH (48 znaków).

ImqString destinationQueueName () const ;

Zwraca kopię **nazwy kolejki docelowej**, która jest pozbawiona końcowych odstępów.

void setDestinationQueueName (const char * nazwa);

Ustawia **nazwę kolejki docelowej**. Obcinanie danych jest dłuższe niż wartość MQ_Q_NAME_LENGTH (48 znaków).

ImqString putApplicationNazwa () const ;

Zwraca kopię **nazwy aplikacji umieszczonej**, która jest pozbawiona końcowych odstępów.

void setPutApplicationName (const char * nazwa = 0);

Ustawia **nazwę umieszczonej aplikacji**. Obcinanie danych jest dłuższe niż wartość MQ_PUT_APPL_NAME_LENGTH (28 znaków).

MQLONG putApplicationTyp () const ;

Zwraca **wstawiony typ aplikacji**.

void setPutApplicationType (const MQLONG typ = MQAT_NO_CONTEXT);

Ustawia **typ aplikacji umieszczonej**.

ImqString putDate () const ;

Zwraca kopię **daty umieszczenia**, która jest pozbawiona końcowych odstępów.

void setPutDate (const char * data = 0);

Ustawia **datę umieszczenia**. Obcinanie danych jest dłuższe niż wartość MQ_PUT_DATE_LENGTH (8 znaków).

ImqString putTime () const ;

Zwraca kopię **czasu umieszczania**, która jest pozbawiona końcowych odstępów.

void setPutTime (const char * czas = 0);

Ustawia **czas umieszczania**. Obcinanie danych jest dłuższe niż wartość MQ_PUT_TIME_LENGTH (8 znaków).

Dane obiektu (chronione)

MQDLH omqdlh

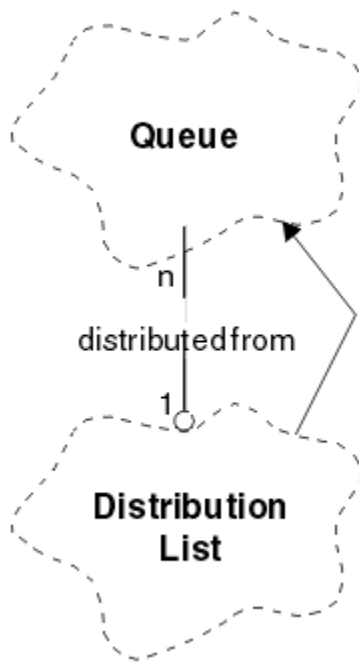
Struktura danych MQDLH.

Kody przyczyny

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_BŁĄD
- Błąd MQRC_ENCODING_ERROR

Klasa ImqDistribution-lista C++

Ta klasa hermetyzuje dynamiczną listę dystrybucyjną, która odwołuje się do jednej lub większej liczby kolejek w celu wysłania komunikatu lub komunikatów do wielu miejsc docelowych.



Rysunek 52. Klasa listy ImqDistribution

- [“Atrybuty obiektu” na stronie 1960](#)
- [“Konstruktory” na stronie 1960](#)
- [“Metody obiektów \(publiczne\)” na stronie 1960](#)
- [“Metody obiektów \(chronione\)” na stronie 1961](#)

Atrybuty obiektu

pierwsza kolejka rozproszona

Pierwszy z jednego lub większej liczby obiektów klasy, w żadnym konkretnym porządku, w którym **odwołanie do listy dystrybucyjnej** odnosi się do tego obiektu.

Początkowo nie ma takich obiektów. Aby pomyślnie utworzyć listę ImqDistribution, musi istnieć co najmniej jeden taki obiekt.

Uwaga: Po otwarciu obiektu listy ImqDistribution wszystkie otwarte obiekty, które odwołują się do niego, są zamykane automatycznie.

Konstruktory

ImqDistributionList ();

Konstruktor domyślny.

Lista ImqDistribution(const ImqDistributionList & lista);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqDistributionList & lista);

Wszystkie obiekty, które odwołują się do **tego** obiektu, są wyłuskane przed kopiowaniem. Żadne obiekty nie będą odwoływać się do **tego** obiektu po wywołaniu tej metody.

* firstDistributedQueue () const ;

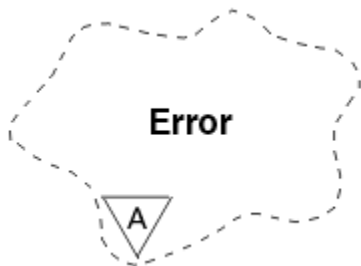
Zwraca **pierwszą kolejkę rozproszoną**.

Metody obiektów (chronione)

void setFirstDistributedQueue (* kolejka = 0);
Ustawia pierwszą kolejkę rozproszoną.

Klasa języka C++ ImqError

Ta klasa abstrakcyjna udostępnia informacje o błędach powiązanych z obiektem.



Rysunek 53. Klasa ImqError

- [“Atrybuty obiektu” na stronie 1961](#)
- [“Konstruktory” na stronie 1961](#)
- [“Metody obiektów \(publiczne\)” na stronie 1961](#)
- [“Metody obiektów \(chronione\)” na stronie 1962](#)
- [“Kody przyczyny” na stronie 1962](#)

Atrybuty obiektu

kod zakończenia

Najnowszy kod zakończenia. Wartością początkową jest zero. Możliwe są następujące wartości dodatkowe:

- MQCC_OK
- MQCC_WARNING,
- MQCC_FAILED

reason code (kod przyczyny)

Najnowszy kod przyczyny. Wartością początkową jest zero.

Konstruktory

ImqError();

Konstruktor domyślny.

ImqError(const ImqError & błqd);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqError & błqd);

Kopiuje dane instancji z *błqd*, zastępując istniejące dane instancji.

void clearErrorCodes ();

Ustawia **kod zakończenia** i **kod przyczyny** zarówno do zera.

MQLONG completionCode () const ;

Zwraca **kod zakończenia**.

MQLONG reasonCode () const ;

Zwraca **kod przyczyny**.

Metody obiektów (chronione)

ImqBoolean checkReadPointer (const void * *wskaznik*, const size_t *długość*);

Sprawdza, czy kombinacja wskaźnika i długości jest poprawna dla dostępu tylko do odczytu, i zwraca wartość PRAWDA, jeśli jest ona pomyślna.

ImqBoolean checkWritePointer (const void * *wskaznik*, const size_t *długość*);

Sprawdza, czy kombinacja wskaźnika i długości jest poprawna w przypadku dostępu do odczytu i zapisu, i zwraca wartość PRAWDA, jeśli jest to pomyślne.

void setCompletionCode (const MQLONG *code* = 0);

Ustawia **kod zakończenia**.

void setReasonCode (const MQLONG *kod* = 0);

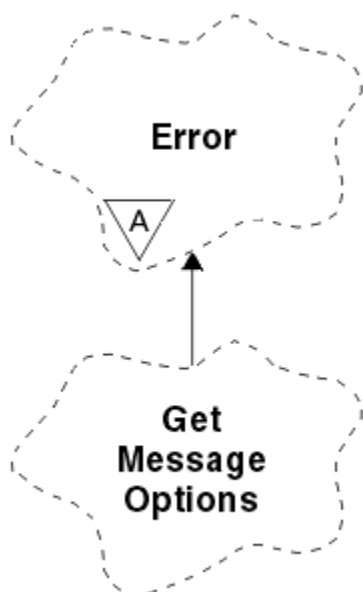
Ustawia **kod przyczyny**.

Kody przyczyny

- MQRC_BUFFER_ERROR-BŁĄD

ImqGetMessageOptions klasa C++

Ta klasa hermetyzuje strukturę danych MQGMO



Rysunek 54. Klasa ImqGetMessageOptions

- [“Atrybuty obiektu” na stronie 1962](#)
- [“Konstruktory” na stronie 1964](#)
- [“Metody obiektów \(publiczne\)” na stronie 1964](#)
- [“Metody obiektów \(chronione\)” na stronie 1965](#)
- [“Dane obiektu \(chronione\)” na stronie 1965](#)
- [“Kody przyczyny” na stronie 1965](#)

Atrybuty obiektu

Status grupy

Status komunikatu dla grupy komunikatów. Wartością początkową jest MQGS_NOT_IN_GROUP. Możliwe są następujące wartości dodatkowe:

- MQGS_MSG_IN_GROUP

- MQGS_LAST_MSG_IN_GROUP

opcje dopasowywania

Opcje wyboru komunikatów przychodzących. Wartością początkową jest MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Możliwe są następujące wartości dodatkowe:

- MQMO_GROUP_ID
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN,
- MQMO_NONE

znacznik komunikatu

Token komunikatu. Wartość binarna (MQBYTE16) o długości MQ_MSG_TOKEN_LENGTH. Wartością początkową jest MQMTOK_NONE.

Opcje

Opcje mające zastosowanie do komunikatu. Wartością początkową jest MQGMO_NO_WAIT. Możliwe są następujące wartości dodatkowe:

- MQGMO_WAIT
- MQGMO_SYNCPOINT,
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- Blokada MQGMO_LOCK
- MQGMO_UNLOCK
- Komunikat 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

rozstrzygnięta nazwa kolejki

Rozstrzygnięta nazwa kolejki. Ten atrybut jest tylko do odczytu. Nazwy nie mogą być dłuższe niż 48 znaków i mogą być dopełniane do tej długości z wartościami pustymi. Wartością początkową jest łańcuch o wartości NULL.

zwrócona długość

Zwracana długość. Wartością początkową jest MQRL_UNDEFINED. Ten atrybut jest tylko do odczytu.

segmentacja

Możliwość segmentowania wiadomości. Wartością początkową jest MQSEG_INHIBITED. Możliwa jest dodatkowa wartość opcji MQSEG_ALLOWED.

status segmentu

Status segmentacji komunikatu. Wartością początkową jest MQSS_NOT_A_SEGMENT. Możliwe są następujące wartości dodatkowe:

- Segment MQSS_SEGMENT
- MQSS_LAST_SEGMENT,

uczestnictwo w punkcie synchronizacji

Wartość TRUE, jeśli komunikaty są pobierane pod kontrolą punktu synchronizacji.

Interwał oczekiwania

Czas, przez jaki metoda **get** klasy jest wstrzymana podczas oczekiwania na nadejście odpowiedniego komunikatu, jeśli nie jest on już dostępny. Wartość początkowa wynosi zero, co powoduje oczekiwanie na czas nieokreślony. Możliwa jest dodatkowa wartość MQWI_UNLIMITED. Ten atrybut jest ignorowany, chyba że parametr **options** zawiera parametr MQGMO_WAIT.

Konstruktory

ImqGetMessageOptions();

Konstruktor domyślny.

ImqGetMessageOptions(const ImqGetMessageOptions & gmo);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqGetMessageOptions & gmo);

Kopiuje dane instancji z *gmo*, zastępując istniejące dane instancji.

MQCHAR groupStatus () const ;

Zwraca **status grupy**.

void setGroupStatus (const MQCHAR status);

Ustawia **status grupy**.

MQLONG matchOptions () const ;

Zwraca **opcje zgodności**.

void setMatchOptions (const MQLONG opcje);

Ustawia **opcje zgodności**.

ImqBinary messageToken() const;

Zwraca **token komunikatu**.

ImqBoolean setMessageToken (const ImqBinary & token);

Ustawia **token komunikatu**. Długość danych elementu *token* musi mieć wartość zero lub wartość MQ_MSG_TOKEN_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setMessageToken (const MQBYTE16 token = 0);

Ustawia **token komunikatu**. *token* może być zerem, który jest taki sam, jak parametr MQMTOK_NONE. Jeśli element *token* ma wartość niezerową, musi być adresowany do bajtów MQ_MSG_TOKEN_LENGTH bajtów danych binarnych.

W przypadku korzystania z predefiniowanych wartości, takich jak MQMTOK_NONE, może nie być konieczne tworzenie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQMTOK_NONE.

MQLONG opcje () const ;

Zwraca **opcje**.

void setOptions (const MQLONG opcje);

Ustawia wartość **options**, w tym wartość **syncpoint participation** .

ImqString resolvedQueueNazwa () const ;

Zwraca kopię **przetłumaczonej nazwy kolejki**.

MQLONG returnedLength() const;

Zwraca **długość zwracaną**.

MQCHAR segmentacja () const ;

Zwraca **segmentację**.

void setSegmentation (const MQCHAR wartość);

Ustawia **segmentację**.

MQCHAR segmentStatus () const ;

Zwraca **status segmentu**.

void setSegmentStatus (const MQCHAR status);

Ustawia **status segmentu**.

ImqBoolean syncPointUdział () const ;

Zwraca wartość **syncpoint participation** , która jest równa TRUE, jeśli **options** to zarówno MQGMO_SYNCPOINT, jak i MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation (const ImqBoolean sync);

Ustawia wartość parametru **syncpoint participation** . Jeśli **sync** ma wartość TRUE, zmienia **opcje** tak, aby obejmował MQGMO_SYNCPOINT i wykluczał zarówno MQGMO_NO_SYNCPOINT, jak i MQGMO_SYNCPOINT_IF_PERSISTENT. Jeśli **sync** ma wartość FALSE, zmienia **opcje** tak, aby zawierała MQGMO_NO_SYNCPOINT i wykluczał zarówno MQGMO_SYNCPOINT, jak i MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval () const ;

Zwraca **przedział czasu oczekiwania**.

void setWaitInterval (const MQLONG interwał);

Ustawia **okres oczekiwania**.

Metody obiektów (chronione)

static void setVersionSupported (const MQLONG);

Ustawia wersję **MQGMO** . Wartość domyślna to **MQGMO_VERSION_3**.

Dane obiektu (chronione)

MQGMO omqgmo

Struktura danych MQGMO w wersji 2. Dostęp do pól MQGMO obsługiwanych tylko w przypadku MQGMO_VERSION_2 .

PMQGMO opgmo

Adres struktury danych MQGMO. Numer wersji dla tego adresu jest wskazany w *olVersion*. Przed uzyskaniem dostępu do pól MQGMO należy sprawdzić numer wersji, aby upewnić się, że są one obecne.

MQLONG olVersion

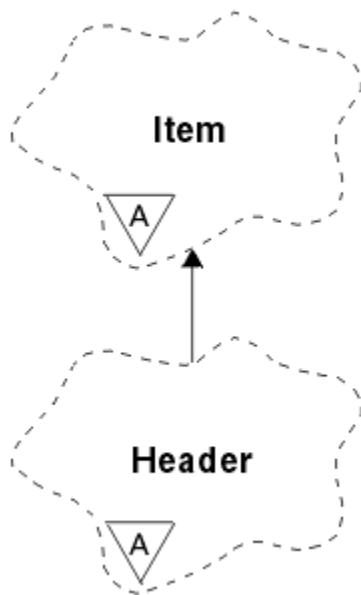
Numer wersji struktury danych MQGMO adresowanej przez *opgmo*.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Klasa języka C++ ImqHeader

Ta klasa abstrakcyjna hermetyzuje wspólne cechy struktury danych MQDLH.



Rysunek 55. Klasa *ImqHeader*

- [“Atrybuty obiektu” na stronie 1966](#)
- [“Konstruktory” na stronie 1966](#)
- [“Metody obiektów \(publiczne\)” na stronie 1966](#)

Atrybuty obiektu

zestaw znaków

Oryginalny kodowany identyfikator zestawu znaków. Początkowo MQCCSI_Q_MGR.

encoding

Oryginalne kodowanie. Początkowo MQENC_NATIVE.

Format

Oryginalny format. Początkowo MQFMT_NONE.

flagi nagłówka

Wartości początkowe to:

- Zero dla obiektów klasy *ImqDeadLetterHeader*
- MQIIH_NONE dla obiektów klasy *ImqIMSBridgeHeader*
- MQRMHF_LAST dla obiektów klasy nagłówka *ImqReference*
- MQCIH_NONE dla obiektów klasy *ImqCICSBridgeHeader*
- MQWIIH_NONE dla obiektów klasy nagłówka *ImqWork*

Konstruktory

ImqHeader();

Konstruktor domyślny.

ImqHeader(const ImqHeader & header);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqHeader & header);

Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

virtual MQLONG characterSet () const ;

Zwraca **zestaw znaków**.

virtual void setCharacterSet (const MQLONG ccsid = MQCCSI_Q_MGR);

Ustawia **zestaw znaków**.

virtual MQLONG kodowanie () const ;

Zwraca **kodowanie**.

virtual void setEncoding (const MQLONG kodowanie = MQENC_NATIVE);

Ustawia **kodowanie**.

wirtualny ImqString format () const ;

Zwraca kopię **formatu**, w tym odstępy końcowe.

wirtualna void setFormat (const char * nazwa = 0);

Ustawia **format**, wyściełany do 8 znaków, ze spacjami kończącymi.

Virtual MQLONG headerFlags () const ;

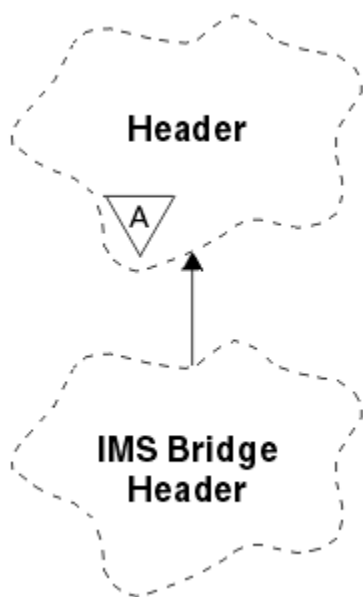
Zwraca **flagi nagłówka**.

virtual void setHeaderFlags (const MQLONG flagi = 0);

Ustawia **flagi nagłówka**.

ImqIMSBridgeHeader C++

Ta klasa hermetyzuje cechy struktury danych MQIIH.



Rysunek 56. Klasa ImqIMSBridgeHeader

Obiekty tej klasy są używane przez aplikacje, które wysyłają komunikaty do mostu IMS za pomocą programu IBM MQ for z/OS.

Uwaga: Wartości ImqHeader **zestaw znaków** i **encoding** muszą mieć wartości domyślne i nie mogą być ustawiane na inne wartości.

- [“Atrybuty obiektu” na stronie 1968](#)
- [“Konstruktor” na stronie 1968](#)
- [“Przeciążone metody ImqItem” na stronie 1968](#)
- [“Metody obiektów \(publiczne\)” na stronie 1968](#)
- [“Dane obiektu \(chronione\)” na stronie 1969](#)
- [“Kody przyczyny” na stronie 1969](#)

Atrybuty obiektu

element uwierzytelniający

RACF (hasło lub passticket) o długości MQ_AUTHENTICATOR_LENGTH. Wartością początkową jest MQIAUT_NONE.

tryb zatwierdzania

Tryb kontroli transakcji. Więcej informacji na temat trybów zatwierdzania produktu IMS zawiera publikacja *OTMA User's Guide*. Wartością początkową jest MQICM_COMMIT_THEN_SEND. Dodatkowa wartość, MQICM_SEND_THEN_COMMIT, jest możliwa.

logiczne nadpisanie terminalu

Nadpisanie terminalu logicznego o długości MQ_LTERM_OVERRIDE_LENGTH. Wartością początkową jest łańcuch o wartości NULL.

nazwa odwzorowania usług w formacie komunikatów

Nazwa odwzorowania MFS o długości MQ_MFS_MAP_NAME_LENGTH. Wartością początkową jest łańcuch o wartości NULL.

format odpowiedzi

Format każdej odpowiedzi o długości MQ_FORMAT_LENGTH. Wartością początkową jest MQFMT_NONE.

zasięg zabezpieczeń

Zasięg przetwarzania zabezpieczeń produktu IMS. Wartością początkową jest MQISS_CHECK. Dodatkowa wartość, MQISS_FULL, jest możliwa.

identyfikator instancji transakcji

Tożsamość instancji transakcji, wartość binarna (MQBYTE16) o długości MQ_TRAN_INSTANCE_ID_LENGTH. Wartością początkową jest MQITII_NONE.

Stan transakcji

Stan konwersacji IMS. Wartością początkową jest MQITS_NOT_IN_CONVERSATION. Dodatkowa wartość, MQITS_IN_CONVERSATION, jest możliwa.

Konstruktory

ImqIMSBridgeHeader();

Konstruktor domyślny.

ImqIMSBridgeHeader(const ImqIMSBridgeHeader & header);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & komunikat);

Wstawia strukturę danych MQIIH do buforu komunikatów na początku i dalej przenosi istniejące dane komunikatu. Ustawia format *msg format* na MQFMT_IMS.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

virtual ImqBoolean pasteIn (ImqMessage & komunikat);

Odczytuje strukturę danych MQIIH z buforu komunikatów.

Aby możliwe było pomyślne, **kodowanie** obiektu *msg* musi mieć wartość MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT na wartość MQENC_NATIVE.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_IMS.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

Metody obiektów (publiczne)

void operator = (const ImqIMSBridgeHeader & nagłówek);

Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqString element uwierzytelniający () const ;

Zwraca kopię elementu **element uwierzytelniający** dopełnione odstępami końcowymi na długości MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator (const char * nazwa);

Ustawia element **authenticator**.

MQCHAR commitMode () const ;

Zwraca **tryb kontroli transakcji**.

void setCommitMode (const MQCHAR tryb);

Ustawia **tryb kontroli transakcji**.

ImqString logicalTerminalOverride () const ;

Zwraca kopię **nadpisanie terminalu logicznego**.

void setLogicalTerminalOverride (const char * override);

Ustawia **logiczne przestąpienie terminalu logicznego**.

ImqString messageFormatServicesMapName () const ;

Zwraca kopię **nazwy odwzorowania usług formatu komunikatów**.

void setMessageFormatServicesMapName (const char * nazwa);

Ustawia **nazwę odwzorowania usług formatu komunikatów**.

ImqString replyToFormat () const ;

Zwraca kopię **formatu odpowiedzi**, dopełnianą odstępami końcowymi do długości MQ_FORMAT_LENGTH.

void setReplyToFormat (const char * format);

Ustawia wartość parametru **reply-to format**, uzupełniając odstępy końcowe na długość MQ_FORMAT_LENGTH.

MQCHAR securityScope () const ;

Zwraca **zasięg zabezpieczeń**.

void setSecurityScope (const MQCHAR zasięg);

Ustawia **zasięg zabezpieczeń**.

ImqBinary transactionInstanceId () const ;

Zwraca kopię **identyfikatora instancji transakcji**.

ImqBoolean setTransactionInstanceId (const ImqBinary & id);

Ustawia **identyfikator instancji transakcji**. Długość danych elementu *token* musi mieć wartość zero lub wartość MQ_TRAN_INSTANCE_ID_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setTransactionInstanceId (const MQBYTE16 id = 0);

Ustawia **identyfikator instancji transakcji**. Wartość *id* może być równa zero, która jest taka sama, jak wartość parametru MQITII_NONE. Jeśli parametr *id* ma wartość niezerową, musi mieć adres MQ_TRAN_INSTANCE_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQITII_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQITII_NONE.

MQCHAR transactionState () const ;

Zwraca **stan transakcji**.

void setTransactionState (const MQCHAR stan);

Ustawia **stan transakcji**.

Dane obiektu (chronione)

MQIIH omqiih

Struktura danych MQIIH.

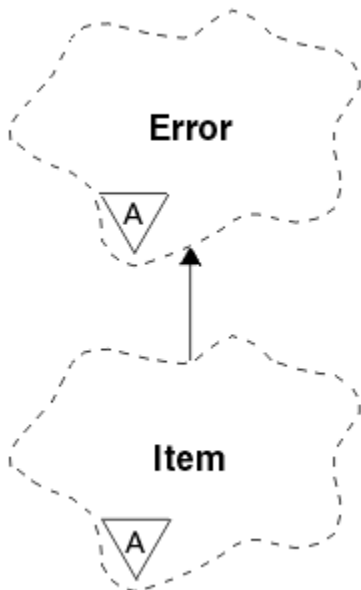
Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

- MQRC_INCONSISTENT_FORMAT
- Błąd MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_BŁĄD

Klasa ImqItem C++

Ta klasa abstrakcyjna reprezentuje element, być może jeden z kilku, w obrębie komunikatu.



Rysunek 57. Klasa ImqItem

Elementy są konkatelowane razem w buforze komunikatów. Każda specjalizacja jest powiązana z określoną strukturą danych, która rozpoczyna się od identyfikatora struktury.

Metody polimorficzne w tej klasie abstrakcyjnej pozwalają na skopiowanie elementów do i z komunikatów. The ImqMessage class **readItem** and **writeItem** methods provide another style of invoking these polymorphic methods that is more natural for application programs.

- [“Atrybuty obiektu” na stronie 1970](#)
- [“Konstruktory” na stronie 1970](#)
- [“Metody klasy \(publiczne\)” na stronie 1971](#)
- [“Metody obiektów \(publiczne\)” na stronie 1971](#)
- [“Kody przyczyny” na stronie 1971](#)

Atrybuty obiektu

identyfikator struktury

Łańcuch zawierający cztery znaki na początku struktury danych. Ten atrybut jest tylko do odczytu. Należy rozważyć ten atrybut dla klas pochodnych. Nie jest on dołączany automatycznie.

Konstruktory

ImqItem();

Konstruktor domyślny.

ImqItem(const ImqItem & pozycja);

Konstruktor kopiowania.

Metody klasy (publiczne)

static ImqBoolean structureIds (const char * *structure-id-to-test*, const ImqMessage & *msg*);

Zwraca wartość PRAWDA, jeśli **identyfikator struktury** następnego elementu ImqItem w przychodzącym *msg* jest taki sam, jak *struktura-id-test*. Następny element jest identyfikowany jako część buforu komunikatów aktualnie adresowanego przez ImqCache **wskaźnik danych**. Ta metoda opiera się na **identyfikatorze struktury** i dlatego nie jest gwarantowana praca dla wszystkich klas pochodnych ImqItem .

Metody obiektów (publiczne)

void operator = (const ImqItem & *pozycja*);

Kopiuje dane instancji z elementu *element*, zastępując istniejące dane instancji.

virtual ImqBoolean copyOut (ImqMessage & *komunikat*) = 0;

Zapisuje ten obiekt jako następny element w wychodzącym buforze komunikatów, dołączając go do wszystkich istniejących elementów. Jeśli operacja zapisu zakończy się pomyślnie, należy zwiększyć ImqCache **długość danych**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Prześlń tę metodę, aby pracować z konkretną podklasą.

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*) = 0;

Odczytuje ten obiekt *destrukcyjnie* z buforu komunikatów przychodzących. Odczyt jest destrukcyjny w tym, że ImqCache **wskaźnik danych** jest przenoszony. Jednak zawartość buforu pozostaje taka sama, więc dane mogą zostać ponownie odczytane przez zresetowanie ImqCache **wskaźnik danych**.

Klasa (pod) tego obiektu musi być spójna z **identyfikatorem struktury** znalezionym obok w buforze komunikatów obiektu *msg* .

Parametr **encoding** obiektu *msg* powinien mieć wartość MQENC_NATIVE. It is recommended that messages be retrieved with the ImqMessage **kodowanie** set to MQENC_NATIVE, and with the ImqGetMessageOptions **opcje** including MQGMO_CONVERT.

Jeśli operacja odczytu zakończy się pomyślnie, wartość ImqCache **długość danych** zostanie zmniejszona. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

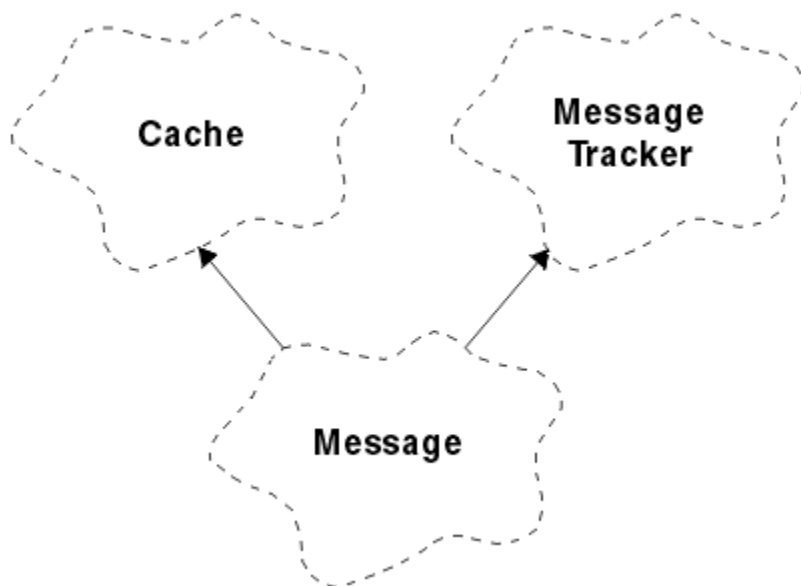
Prześlń tę metodę, aby pracować z konkretną podklasą.

Kody przyczyny

- Błąd MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_BŁĄD
- MQRC_INCONSISTENT_FORMAT
- MQRC_INSUFFICIENT_BUFFER
- MQRC_INSUFFICIENT_DATA

Klasa języka C++ ImqMessage

Klasa ta hermetyzuje strukturę danych MQMD, a także zajmuje się budową i odbudową danych komunikatu.



Rysunek 58. Klasa *ImqMessage*

- [“Atrybuty obiektu” na stronie 1972](#)
- [“Konstruktory” na stronie 1976](#)
- [“Metody obiektów \(publiczne\)” na stronie 1976](#)
- [“Metody obiektów \(chronione\)” na stronie 1978](#)
- [“Dane obiektu \(chronione\)” na stronie 1978](#)

Atrybuty obiektu

Informacje identyfikujące aplikację

Informacje o tożsamości powiązane z komunikatem. Wartością początkową jest łańcuch o wartości NULL.

Dane pochodzenia aplikacji

Informacje o pochodzeniu powiązane z komunikatem. Wartością początkową jest łańcuch o wartości NULL.

Licznik wycofań

Liczba przypadków, w których komunikat został wstępnie pobrany, a następnie wycofany. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

zestaw znaków

Identyfikator kodowanego zestawu znaków. Wartością początkową jest MQCCSI_Q_MGR. Możliwe są następujące wartości dodatkowe:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

Do wyboru można również użyć identyfikatora kodowanego zestawu znaków. Więcej informacji na ten temat zawiera sekcja [“konwersja stron kodowych” na stronie 951](#).

encoding

Kodowanie maszynowe danych komunikatu. Wartością początkową jest MQENC_NATIVE.

Utrata ważności

Zależna od czasu ilość, która kontroluje, jak długo IBM MQ zachowuje nieodzyskany komunikat przed usunięciem go. Wartością początkową jest MQEI_UNLIMITED.

Format

Nazwa formatu (szablonu), który opisuje układ danych w buforze. Nazwy dłuższe niż osiem znaków są obcinane do ośmiu znaków. Nazwy są zawsze dopełniane spacjami do ośmiu znaków. Początkowa wartość stała to MQFMT_NONE. Możliwe są następujące dodatkowe stałe:

- ADMINISTRATOR MQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- Zdarzenie MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- Nagłówek MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

Można również użyć wybranego przez użytkownika łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat można znaleźć w polu [“Format \(MQCHAR8\)” na stronie 433](#) deskryptora komunikatu (MQMD).

Flagi komunikatu

Informacje sterujące segmentacją. Wartością początkową jest MQMF_SEGMENTATION_INHIBITED. Możliwe są następujące wartości dodatkowe:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE

typ komunikatu

Szeroka kategoryzacja komunikatu. Wartością początkową jest MQMT_DATAGRAM. Możliwe są następujące wartości dodatkowe:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- Raport_menedżera_mQMT
- MQMT_APPL_FIRST
- MQMT_APPL_LAST

Można również użyć wybranej wartości specyficznej dla aplikacji. Więcej informacji na ten temat można znaleźć w polu `MsgType (MQLONG)` na stronie 445 deskryptora komunikatu (MQMD).

Przesunięcie

Przesunięcie informacji. Wartością początkową jest zero.

Pierwotna długość

Oryginalna długość segmentowanego komunikatu. Wartością początkową jest `MQOL_UNDEFINED`.

trwałość

Wskazuje, że komunikat jest ważny i musi być w każdym momencie składowany przy użyciu trwałej pamięci masowej. Ta opcja wiąże się z karą wydajności. Wartością początkową jest `MQPER_PERSISTENCE_AS_Q_DEF`. Możliwe są następujące wartości dodatkowe:

- `MQPER_PERSISTENT`
- `MQPER_NOT_PERSISTENT`

priorytet

Względny priorytet przesyłania i dostarczania. Komunikaty o tym samym priorytecie są zwykle dostarczane w tej samej kolejności, w jakiej zostały dostarczone (choć istnieje kilka kryteriów, które muszą być spełnione, aby zagwarantować tę gwarancję). Wartością początkową jest `MQPRI_PRIORITY_AS_Q_DEF`.

sprawdzanie poprawności właściwości

Określa, czy sprawdzanie poprawności właściwości powinno mieć miejsce po ustawieniu właściwości komunikatu. Wartością początkową jest `MQCMHO_DEFAULT_VALIDATION`. Możliwe są następujące wartości dodatkowe:

- `MQCMHO_VALIDATE`
- `MQCMHO_NO_VALIDATION`

W przypadku **sprawdzania poprawności właściwości** działają następujące metody:

`MQLONG propertyValidation() const;`

Zwraca opcję **sprawdzania poprawności właściwości**.

`void setPropertyValidation (const MQLONG opcja );`

Ustawia opcję **sprawdzania poprawności właściwości**.

Nazwa aplikacji wstawiającej

Nazwa aplikacji, która umiała komunikat. Wartością początkową jest łańcuch o wartości `NULL`.

Typ aplikacji wstawiającej

Typ aplikacji, która wstawiła komunikat. Wartością początkową jest `MQAT_NO_CONTEXT`. Możliwe są następujące wartości dodatkowe:

- `MQAT_AIX`
- `MQAT_CICS`
- `MQAT_CICS_BRIDGE`
- `MQAT_DOS`
- `MQAT_IMS`
- `MQAT_IMS_BRIDGE`
- `MQAT_MVS`
- `MQAT_NOTES_AGENT`
- `MQAT_OS2`
- `MQAT_OS390`
- `MQAT_OS400`
- `MQAT_QMGR`
- `MQAT_UNIX`
- `MQAT_WINDOWS`

- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT
- MQAT_UNKNOWN
- MQAT_USER_FIRST
- MQAT_USER_LAST

Można również użyć wybranego przez użytkownika łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat można znaleźć w polu [“Typ PutAppl\(MQLONG\)”](#) na stronie 450 deskryptora komunikatu (MQMD).

Data wstawienia

Data umieszczenia komunikatu. Wartością początkową jest łańcuch o wartości NULL.

Czas wstawienia

Godzina umieszczenia komunikatu. Wartością początkową jest łańcuch o wartości NULL.

nazwa menedżera kolejek odpowiedzi

Nazwa menedżera kolejek, do którego powinna zostać wysłana odpowiedź. Wartością początkową jest łańcuch o wartości NULL.

nazwa kolejki odpowiedzi

Nazwa kolejki, do której powinna zostać wysłana odpowiedź. Wartością początkową jest łańcuch o wartości NULL.

report

Informacje zwrotne powiązane z komunikatem. Wartością początkową jest MQRO_NONE. Możliwe są następujące wartości dodatkowe:

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_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 (Identyfikator CORREL_ID)
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

gdzie * wskazuje wartości, które nie są obsługiwane w produkcie IBM MQ for z/OS.

numer kolejny

Informacje o sekwencji identyfikujące komunikat w grupie. Wartością początkową jest jeden.

całkowita długość komunikatu

Liczba bajtów, które były dostępne podczas ostatniej próby odczytania komunikatu. Liczba ta będzie większa niż `ImqCache` **długość komunikatu** , jeśli ostatni komunikat został obcięty, lub jeśli ostatni komunikat nie został odczytany, ponieważ nastąpiło obcięcie. Ten atrybut jest tylko do odczytu. Wartością początkową jest zero.

Ten atrybut może być przydatny w każdej sytuacji obejmującej obcięte komunikaty.

ID użytkownika

Tożsamość użytkownika powiązana z komunikatem. Wartością początkową jest łańcuch o wartości NULL.

Konstruktory

`ImqMessage()`;

Konstruktor domyślny.

`ImqMessage( const ImqMessage & komunikat )`;

Konstruktor kopiowania. Szczegółowe informacje można znaleźć w metodzie **`operator =`** .

Metody obiektów (publiczne)

`void operator = ( const ImqMessage & komunikat )`;

Kopiuje dane MQMD i dane komunikatu z `msg`. Jeśli użytkownik dla tego obiektu dostarczył bufor, ilość skopiowanych danych jest ograniczona do dostępnej wielkości buforu. W przeciwnym razie system zapewnia, że bufor o odpowiedniej wielkości zostanie udostępniony dla skopiowanych danych.

`ImqString applicationIdData () const` ;

Zwraca kopię **danych identyfikatora aplikacji**.

`void setApplicationIdData ( const char * dane = 0)`;

Ustawia **dane identyfikatora aplikacji**.

`ImqString applicationOriginData () const` ;

Zwraca kopię **danych o pochodzeniu aplikacji**.

`void setApplicationOriginData ( const char * dane = 0)`;

Ustawia **dane o pochodzeniu aplikacji**.

`MQLONG backoutCount () const` ;

Zwraca **liczbę wycofań**.

`MQLONG characterSet () const` ;

Zwraca **zestaw znaków**.

`void setCharacterSet ( const MQLONG ccsid = MQCCSI_Q_MGR)`;

Ustawia **zestaw znaków**.

`MQLONG kodowanie () const` ;

Zwraca **kodowanie**.

`void setEncoding ( const MQLONG kodowanie = MQENC_NATIVE)`;

Ustawia **kodowanie**.

`MQLONG wygaśnięcie () const` ;

Zwraca **wygaśnięcie**.

`void setExpiry ( const MQLONG utrata ważności )`;

Ustawia **wygaśnięcie**.

`ImqString format () const` ;

Zwraca kopię **formatu**, w tym odstępy końcowe.

`ImqBoolean formatIs ( const char * format-do-test ) const` ;

Zwraca TRUE, jeśli **format** jest taki sam jak *format-do-test*.

`void setFormat ( const char * nazwa = 0)`;

Służy do ustawiania **formatu**, dopełnionego do ośmiu znaków, z odstępami końcowymi.

MQLONG messageFlags () const ;

Zwraca **flagi komunikatu**.

void setMessageFlags (const MQLONG *flagi*);

Ustawia **flagi komunikatu**.

MQLONG messageType () const ;

Zwraca **typ komunikatu**.

void setMessageType (const MQLONG *typ*);

Ustawia **typ komunikatu**.

MQLONG przesunięcie () const ;

Zwraca wartość **przesunięcie**.

void setOffset (const MQLONG *przesunięcie*);

Ustawia **przesunięcie**.

MQLONG originalLength () const ;

Zwraca **oryginalną długość**.

void setOriginalLength (const MQLONG *długość*);

Ustawia **oryginalną długość**.

MQLONG trwałość () const ;

Zwraca wartość **trwałość**.

void setPersistence (const MQLONG *trwałość*);

Ustawia **trwałość**.

MQLONG priorytet () const ;

Zwraca **priorytet**.

void setPriority (const MQLONG *priorytet*);

Ustawia **priorytet**.

ImqString putApplicationNazwa () const ;

Zwraca kopię **nazwy aplikacji umieszczonej**.

void setPutApplicationName (const char * *nazwa* = 0);

Ustawia **nazwę umieszczonej aplikacji**.

MQLONG putApplicationTyp () const ;

Zwraca **wstawiony typ aplikacji**.

void setPutApplicationType (const MQLONG *typ* = MQAT_NO_CONTEXT);

Ustawia **typ aplikacji umieszczonej**.

ImqString putDate () const ;

Zwraca kopię **daty umieszczenia**.

void setPutDate (const char * *data* = 0);

Ustawia **datę umieszczenia**.

ImqString putTime () const ;

Zwraca kopię **czasu umieszczenia**.

void setPutTime (const char * *czas* = 0);

Ustawia **czas umieszczania**.

ImqBoolean readItem (ImqItem & *pozycja*);

Odczytuje do obiektu *element* z buforu komunikatów, używając metody ImqItem **pasteIn** . Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString replyToQueueManagerNazwa () const ;

Zwraca kopię **nazwy menedżera kolejek zwrotnych do kolejki**.

void setReplyToQueueManagerName (const char * *nazwa* = 0);

Ustawia **odpowiedź-na nazwę menedżera kolejek**.

ImqString replyToQueueName () const ;

Zwraca kopię **nazwy kolejki odpowiedzi**.

void setReplyToQueueName (const char * *nazwa* = 0);

Ustawia **nazwę** kolejki odpowiedzi.

MQLONG raport () const ;

Zwraca **raport**.

void setReport (const MQLONG *raport*);

Ustawia **raport**.

MQLONG sequenceNumber () const ;

Zwraca **numer kolejny**.

void setSequenceNumber (const MQLONG *liczba*);

Ustawia **numer kolejny**.

size_t totalMessageLength () const ;

Zwraca **łączną długość komunikatu**.

ImqString userId () const ;

Zwraca kopię **ID użytkownika**.

void setUserId (const char * *id* = 0);

Ustawia **ID użytkownika**.

ImqBoolean writeItem (ImqItem & *pozycja*);

Zapisuje dane z obiektu *element* do buforu komunikatów przy użyciu metody ImqItem **copyOut** . Zapis może przybrać formę wstawiania, zastępowania lub dopisywania: zależy to od klasy obiektu *element* .

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Metody obiektów (chronione)

static void setVersionSupported (const MQLONG);

Ustawia **wersję MQMD**. Wartość domyślna to **MQMD_VERSION_2**.

Dane obiektu (chronione)

MQMD1 *komenda omqmd*

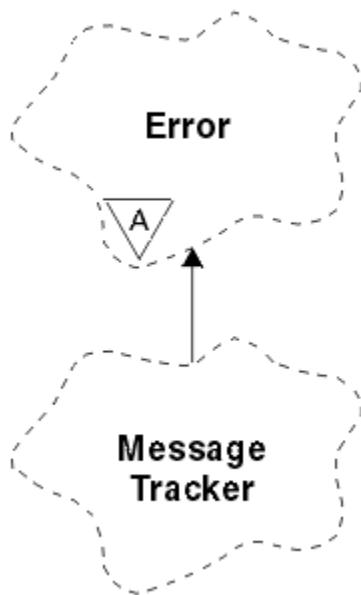
(tylko IBM MQ for z/OS). Struktura danych MQMD.

MQMD2 *omqmd*

(Platformy inne niż z/OS). Struktura danych MQMD.

Klasa ImqMessageTracker C++

Ta klasa hermetykuje te atrybuty obiektu ImqMessage lub ImqQueue , które mogą być powiązane z jednym z obiektów.



Rysunek 59. Klasa Tracker ImqMessage

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“ImqMessage-odniesienie do programu śledzący”](#) na stronie 1928.

- [“Atrybuty obiektu”](#) na stronie 1979
- [“Konstruktory”](#) na stronie 1980
- [“Metody obiektów \(publiczne\)”](#) na stronie 1980
- [“Kody przyczyny”](#) na stronie 1981

Atrybuty obiektu

Token rozliczania

Wartość binarna (MQBYTE32) o długości MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową jest MQACT_NONE.

Identyfikator korelacji

Wartość binarna (MQBYTE24) o długości MQ_CORREL_ID_LENGTH, która jest przypisana do korelowania komunikatów. Wartością początkową jest MQCI_NONE. Możliwa jest dodatkowa wartość MQCI_NEW_SESSION.

Feedback

Informacja zwrotna do wysłania z komunikatem. Wartością początkową jest MQFB_NONE. Możliwe są następujące wartości dodatkowe:

- MQFB_SYSTEM_FIRST
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- MQFB_APPL_LAST
- MQFB_COA
- MQFB_COD
- MQFB_EXPIRATION
- MQFB_PAN
- MQFB_NAN
- MQFB_QUIT
- MQFB_DATA_LENGTH_ZERO

- MQFB_DŁUGOŚĆ_DŁUGOŚĆ_UJEMNEGO
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- BŁĄD MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IMS
- BŁĄD MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_NIEPOWODZENIE bridge_failure
- MQFB_CICS-BŁĄD ID_CCSID
- MQFB_CICS-BŁĄD CIH_ERROR
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_CORREL_ID_ERROR
- MQFB_CICS_DLQ_ERROR
- MQFB_CICSBŁĄD __ENCODING_ERROR
- MQFB_CICS_INTERNAL_ERROR
- MQFB_CICS_NOT_AUTHORIZED
- MQFB_CICS_UOW_BACKED_OUT
- MQFB_CICS_UOW_ERROR

Można również użyć wybranego przez użytkownika łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat można znaleźć w polu [“Opinia \(MQLONG\)”](#) na stronie 430 deskryptora komunikatu (MQMD).

Identyfikator grupy

Wartość binarna (MQBYTE24) o długości MQ_GROUP_ID_LENGTH unikalna w obrębie kolejki. Wartością początkową jest MQGI_NONE.

Identyfikator komunikatu

Wartość binarna (MQBYTE24) o długości MQ_MSG_ID_LENGTH unikalna w obrębie kolejki. Wartością początkową jest MQMI_NONE.

Konstruktory

ImqMessageTracker ();

Konstruktor domyślny.

ImqMessageTracker (const ImqMessageTracker & tracker);

Konstruktor kopiowania. Szczegółowe informacje można znaleźć w metodzie **operator =**.

Metody obiektów (publiczne)

void operator = (const ImqMessageTracker & tracker);

Kopiuje dane instancji z *tracker*, zastępując istniejące dane instancji.

ImqBinary accountingToken () const ;

Zwraca kopię **znacznika rozliczeniowego**.

ImqBoolean setAccountingToken (const ImqBinary & token);

Ustawia **token rozliczania**. Długość danych elementu *token* musi mieć wartość zero lub MQ_ACCOUNTING_TOKEN_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setAccountingToken (const MQBYTE32 token = 0);

Ustawia **token rozliczania**. *token* może być zerem, który jest taki sam, jak parametr MQACT_NONE. Jeśli element *token* ma wartość niezerową, musi zwracać wartość MQ_ACCOUNTING_TOKEN_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQACT_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQACT_NONE.

ImqBinary correlationId () const ;

Zwraca kopię **identyfikatora korelacji**.

ImqBoolean setCorrelationId (const ImqBinary & token);

Ustawia **identyfikator korelacji**. **Długość danych** elementu *token* musi mieć wartość zero lub wartość MQ_CORREL_ID_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setCorrelationId (const MQBYTE24 id = 0);

Ustawia **identyfikator korelacji**. Wartość *id* może być równa zero, która jest taka sama, jak wartość parametru MQCI_NONE. Jeśli parametr *id* ma wartość niezerową, musi mieć adres MQ_CORREL_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQCI_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQCI_NONE.

MQLONG opinia () const ;

Zwraca **informację zwrotną**.

void setFeedback (const MQLONG feedback);

Ustawia **informację zwrotną**.

ImqBinary groupId () const ;

Zwraca kopię **identyfikatora grupy**.

ImqBoolean setGroupId (const ImqBinary & token);

Ustawia **identyfikator grupy**. **Długość danych** elementu *element* musi mieć wartość zero lub wartość MQ_GROUP_ID_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setGroupId (const MQBYTE24 id = 0);

Ustawia **identyfikator grupy**. Wartość *id* może być równa zero, która jest taka sama, jak określenie parametru MQGI_NONE. Jeśli parametr *id* ma wartość niezerową, musi mieć adres MQ_GROUP_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQGI_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQGI_NONE.

ImqBinary messageId () const ;

Zwraca kopię komunikatu **ID komunikatu**.

ImqBoolean setMessageId (const ImqBinary & token);

Ustawia **identyfikator komunikatu**. **Długość danych** elementu *token* musi mieć wartość zero lub wartość MQ_MSG_ID_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setMessageId (const MQBYTE24 id = 0);

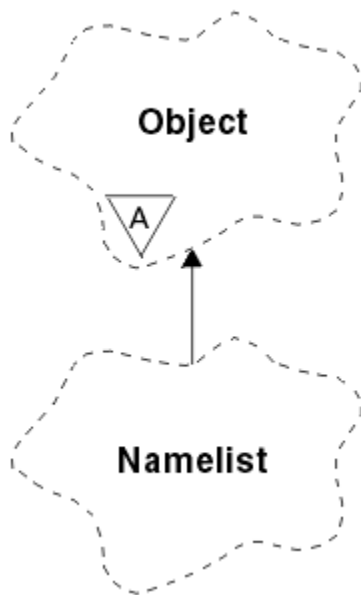
Ustawia **identyfikator komunikatu**. Wartość *id* może być równa zero, która jest taka sama, jak podana wartość MQMI_NONE. Jeśli parametr *id* ma wartość niezerową, musi mieć adres MQ_MSG_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQMI_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQMI_NONE.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Klasa ImqNamelist C++

Ta klasa hermetyzuje listę nazw.



Rysunek 60. Klasa ImqNamelist

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“Odwołanie do tabeli ImqNamelist”](#) na stronie 1928.

- [“Atrybuty obiektu”](#) na stronie 1982
- [“Konstruktory”](#) na stronie 1982
- [“Metody obiektów \(publiczne\)”](#) na stronie 1982
- [“Kody przyczyny”](#) na stronie 1983

Atrybuty obiektu

Liczba nazw

Liczba nazw obiektów w **nazwach list nazw**. Ten atrybut jest tylko do odczytu.

nazwy list nazw

Nazwy obiektów, których liczba jest wskazywana przez **liczbę nazw**. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqNamelist();

Konstruktor domyślny.

ImqNamelist(const ImqNamelist & lista);

Konstruktor kopiowania. Obiekt ImqObject **open status** ma wartość false.

ImqNamelist(const char * nazwa);

Ustawia nazwę ImqObject na **name**.

Metody obiektów (publiczne)

void operator = (const ImqNamelist & lista);

Kopiuje dane instancji z listy *lista*, zastępując istniejące dane instancji. Obiekt ImqObject **open status** ma wartość false.

ImqBoolean nameCount(MQLONG & liczba);

Udostępnia kopię **liczby nazw**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG nameCount ();

Zwraca **liczbę nazw** bez wskazania możliwych błędów.

ImqBoolean namelistName (const MQLONG indeks, ImqString & nazwa);

Udostępnia kopię jednej z **nazw list nazw** według indeksu zerowego. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString namelistName (const MQLONG indeks);

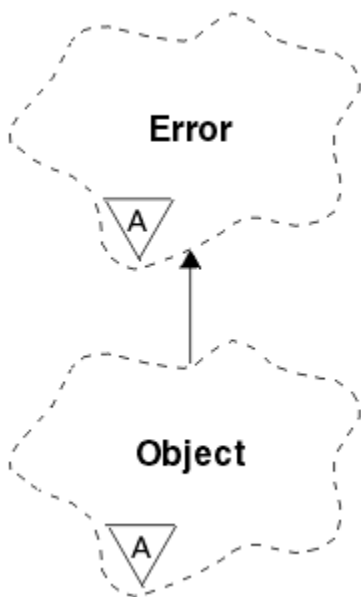
Zwraca jedną z **nazw list nazw** według indeksu zerowego bez wskazania możliwych błędów.

Kody przyczyny

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

Klasa języka C++ ImqObject

Ta klasa jest abstrakcyjna. Jeśli obiekt tej klasy zostanie zniszczony, zostanie on automatycznie zamknięty, a jego połączenie z menedżerem ImqQueue zostanie zerwane.



Rysunek 61. Klasa ImqObject

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“Odwołanie krzyżowe ImqObject”](#) na stronie 1928.

- [“Atrybuty klasy”](#) na stronie 1983
- [“Atrybuty obiektu”](#) na stronie 1984
- [“Konstruktory”](#) na stronie 1985
- [“Metody klasy \(publiczne\)”](#) na stronie 1985
- [“Metody obiektów \(publiczne\)”](#) na stronie 1985
- [“Metody obiektów \(chronione\)”](#) na stronie 1987
- [“Dane obiektu \(chronione\)”](#) na stronie 1988
- [“Kody przyczyny”](#) na stronie 1988
-

Atrybuty klasy

zachowanie

Steruje zachowaniem niejawnego otwierania.

IMQ_IMPL_OPEN (8L)

Dozwolone jest otwarcie niejawne. Jest to opcja domyślna.

Atrybuty obiektu

Data zmiany

Data zmiany. Ten atrybut jest tylko do odczytu.

Godzina zmiany

Godzina zmiany. Ten atrybut jest tylko do odczytu.

Alternatywne ID użytkownika

Alternatywny identyfikator użytkownika, maksymalnie do wartości MQ_USER_ID_LENGTH. Wartością początkową jest łańcuch o wartości NULL.

alternatywny identyfikator zabezpieczeń

Alternatywny identyfikator zabezpieczeń. Wartość binarna (MQBYTE40) o długości MQ_SECURITY_ID_LENGTH. Wartością początkową jest MQSID_NONE.

opcje zamknięcia

Opcje, które mają zastosowanie w przypadku zamknięcia obiektu. Wartością początkową jest MQCO_NONE. Ten atrybut jest ignorowany podczas niejawnych operacji ponownego otwarcia, gdzie zawsze używana jest wartość MQCO_NONE.

połączenie odniesienia

Odwołanie do obiektu menedżera kolejek ImqQueue, który udostępnia wymagane połączenie z menedżerem kolejek (lokalnym). W przypadku obiektu menedżera ImqQueue jest to sam obiekt. Wartością początkową jest zero.

Uwaga: Nie należy go mylić z **nazwą menedżera kolejek**, która identyfikuje menedżer kolejek (być może zdalny) dla określonej kolejki.

Opis

Nazwa opisowa (maksymalnie 64 znaki) menedżera kolejek, kolejki, listy nazw lub procesu. Ten atrybut jest tylko do odczytu.

Nazwa

Nazwa (o długości do 48 znaków) menedżera kolejek, kolejki, listy nazw lub procesu. Wartością początkową jest łańcuch o wartości NULL. Nazwa kolejki modelowej zmienia się po **otwarcu** na nazwę wynikowej kolejki dynamicznej.

Uwaga: Menedżer kolejek ImqQueue może mieć pustą nazwę reprezentującą domyślny menedżer kolejek. Nazwa zostanie zmieniona na rzeczywisty menedżer kolejek po pomyślnym **otwarcu**. Lista ImqDistribution jest dynamiczna i musi mieć nazwę o wartości NULL.

następny obiekt zarządzany

Jest to następny obiekt tej klasy, w żadnym konkretnym porządku, o tym samym **odwołaniu do połączenia**, który jest obiektem tego obiektu. Wartością początkową jest zero.

Opcje otwarcia

Opcje, które mają zastosowanie podczas otwierania obiektu. Wartością początkową jest MQOO_INQUIRE. Istnieją dwa sposoby ustawiania odpowiednich wartości:

1. Nie należy ustawiać **opcji otwierania** i nie należy używać metody **open**. Program IBM MQ automatycznie dostosowuje **opcje otwierania** i automatycznie otwiera, ponownie otwiera i zamyka obiekty zgodnie z wymaganiami. Może to spowodować niepotrzebne operacje ponownego otwarcia, ponieważ produkt IBM MQ korzysta z metody **openFor**, co powoduje, że opcja **open options** jest dodawana tylko przyrostowo.
2. Przed użyciem metod, których wynikiem jest wywołanie MQI, należy ustawić **opcje otwarcia** (patrz "Skorowidz języka C++ i MQI" na stronie 1922). Zapewnia to, że zbędne operacje ponownego otwarcia nie zostaną wykonane. Ustaw opcje otwierania jawnie, jeśli prawdopodobne jest wystąpienie potencjalnych problemów z ponownym otwartymi otwartymi opcjami (patrz sekcja Otwórz ponownie).

Jeśli używana jest metoda **open** , użytkownik *musi* upewnić się, że **opcje otwarcia** są odpowiednie jako pierwsze. Jednak użycie metody **open** nie jest obowiązkowe; produkt IBM MQ nadal wykazuje takie samo zachowanie jak w przypadku 1, ale w tym przypadku zachowanie jest wydajne.

Wartość zero nie jest poprawną wartością. Należy ustawić odpowiednią wartość przed próbą otwarcia obiektu. Można to zrobić za pomocą programu **Opcje programu setOpen** (*Opcje programu lOpen*) po którym następuje **otwarte** () lub **openFor** (*IRequiredOpenOption*).

Uwaga:

1. W metodzie MQOO_INQUIRE w metodzie **open** dla listy dystrybucyjnej jest podstawiana MQOO_OUTPUT, ponieważ parametr MQOO_OUTPUT jest w tym momencie jedyną poprawną opcją **open** . Zaleca się jednak, aby w programach aplikacji, które używają metody **open** , jawnie ustawić parametr MQOO_OUTPUT.
2. Określ atrybuty MQOO_RESOLVE_NAMES, jeśli mają być używane atrybuty **przetłumaczone nazwy menedżera kolejek** i **rozstrzygnięte nazwy kolejki** klasy.

status otwarcia

Określa, czy obiekt jest otwarty (TRUE), czy zamknięty (FALSE). Wartością początkową jest FALSE. Ten atrybut jest tylko do odczytu.

poprzedni obiekt zarządzany

Poprzedni obiekt tej klasy, w żadnym konkretnym zamówieniu, o tym samym **odwołaniu do połączenia** , co ten obiekt. Wartością początkową jest zero.

identyfikator menedżera kolejek

Identyfikator menedżera kolejek. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqObject();

Konstruktor domyślny.

ImqObject(const ImqObject & obiekt);

Konstruktor kopiowania. **Status otwarcia** ma wartość FAŁSZ.

Metody klasy (publiczne)

statyczne zachowanie MQLONG ();

Zwraca **zachowanie**.

void setBehavior(const MQLONG zachowanie = 0);

Ustawia **zachowanie**.

Metody obiektów (publiczne)

void operator = (const ImqObject & obiekt);

Wykonuje zamknięcie, jeśli jest to konieczne, i kopiuje dane instancji z obiektu *obiekt*. **Status otwarcia** ma wartość FAŁSZ.

ImqBoolean alterationDate(ImqString & data);

Udostępnia kopię **daty zmiany**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString alterationDate();

Zwraca **datę zmiany** bez wskazania ewentualnych błędów.

ImqBoolean alterationTime(ImqString & czas);

Udostępnia kopię **czasu zmiany**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString alterationTime();

Zwraca **czas zmiany** bez wskazania ewentualnych błędów.

ImqString alternateUserId () const ;

Zwraca kopię **alternatywnego identyfikatora użytkownika**.

ImqBoolean setAlternateUserId (const char * id);

Ustawia **alternatywny identyfikator użytkownika**. **alternatywny identyfikator użytkownika** można ustawić tylko wtedy, gdy **status otwarcia** ma wartość FAŁSZ. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBinary alternateSecurityId () const;

Zwraca kopię **alternatywnego identyfikatora zabezpieczeń** .

ImqBoolean setAlternateSecurityId(const ImqBinary & token);

Ustawia **alternatywny identyfikator zabezpieczeń**. **Alternatywny identyfikator zabezpieczeń** można ustawić tylko wtedy, gdy **otwarty status** ma wartość FALSE. Długość danych *token* musi mieć wartość zerową lub MQ_SECURITY_ID_LENGTH. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setAlternateSecurityId(const MQBYTE* token = 0);

Ustawia **alternatywny identyfikator zabezpieczeń**. *token* może być zerem, który jest taki sam, jak parametr MQSID_NONE. Jeśli element *token* ma wartość niezerową, musi mieć adres MQ_SECURITY_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQSID_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQSID_NONE.

Alternatywny identyfikator zabezpieczeń można ustawić tylko wtedy, gdy **otwarty status** ma wartość TRUE. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setAlternateSecurityId(const unsigned char * id = 0);

Ustawia **alternatywny identyfikator zabezpieczeń**.

ImqBoolean close ();

Ustawia **status otwarcia** na FALSE. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG closeOptions () const ;

Zwraca **opcje zamknięcia**.

void setCloseOptions (const MQLONG opcje);

Ustawia **opcje zamknięcia**.

ImqQueueManager * connectionReference () const ;

Zwraca **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager & menedżer);

Ustawia **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager * manager = 0);

Ustawia **odwołanie do połączenia**.

virtual ImqBoolean opis (ImqString & opis) = 0;

Udostępnia kopię **opisu**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString opis ();

Zwraca kopię **opisu** bez wskazania ewentualnych błędów.

virtual ImqBoolean nazwa (ImqString & nazwa);

Udostępnia kopię **nazwy**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString nazwa ();

Zwraca kopię **nazwy** bez wskazania ewentualnych błędów.

ImqBoolean setName (const char * nazwa = 0);

Ustawia **nazwę**. Parametr **name** można ustawić tylko wtedy, gdy **open status** ma wartość FALSE, a dla menedżera ImqQueue, natomiast **status połączenia** ma wartość FALSE. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqObject * nextManagedObject () const ;

Zwraca **następny obiekt zarządzany**.

ImqBoolean open ();

Zmienia wartość parametru **open status** na TRUE, otwierając obiekt w razie potrzeby, korzystając między innymi z atrybutów **open options** i **name**. Ta metoda korzysta z informacji **odwołania połączenia** i metody **connect** menedżera ImqQueue, jeśli jest to konieczne, aby mieć pewność, że **status połączenia** menedżera ImqQueue ma wartość TRUE. Zwraca **status otwarcia**.

ImqBoolean openFor (const MQLONG *wymagane-opcje* = 0);

Próbuje się upewnić, że obiekt jest otwarty z opcjami **open options** lub **open options**, które gwarantują zachowanie implikowane przez wartość parametru *required-options*.

Jeśli parametr *required-options* ma wartość zero, dane wejściowe są wymagane, a wszystkie opcje wejściowe są wystarczające. Tak więc, jeśli **opcje otwarcia** zawierają już jedną z następujących wartości:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

Opcje otwarcia są już zadowalające i nie są zmieniane; jeśli **opcje otwarcia** nie zawierają już żadnej z tych opcji, opcja MQOO_INPUT_AS_Q_DEF jest ustawiona w **otwartych opcjach**.

Jeśli *required-options* ma wartość niezerową, wymagane opcje są dodawane do **opcji otwartych**; Jeśli *wymagane-opcje* to dowolna z tych opcji, pozostałe są resetowane.

Jeśli którekolwiek z **otwartych opcji** zostanie zmienione, a obiekt jest już otwarty, obiekt jest tymczasowo zamknięty i ponownie otwarty w celu dostosowania **otwartych opcji**.

Zwraca wartość PRAWDA, jeśli powiodła się. Powodzenie wskazuje, że obiekt jest otwarty z odpowiednimi opcjami.

MQLONG openOptions () const ;

Zwraca **otwarte opcje**.

ImqBoolean setOpenOpcje (const MQLONG *opcje*);

Ustawia **opcje otwarcia**. **Opcje otwarcia** można ustawić tylko wtedy, gdy **otwarty status** ma wartość FAŁSZ. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean openStatus () const ;

Zwraca **status otwarcia**.

ImqObject * previousManagedObject () const ;

Zwraca **poprzedni obiekt zarządzany**.

ImqBoolean queueManagerIdentyfikator (ImqString & *id*);

Udostępnia kopię **identyfikatora menedżera kolejek**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString queueManagerIdentyfikator ();

Zwraca **identyfikator menedżera kolejek** bez wskazania ewentualnych błędów.

Metody obiektów (chronione)**virtual ImqBoolean closeTemporarily ();**

Zamyka obiekt bezpiecznie przed ponownym otwarciem obiektu. Zwraca wartość PRAWDA, jeśli powiodła się. W tej metodzie założono, że **status otwarcia** ma wartość PRAWDA.

MQHCONN connectionHandle () const ;

Zwraca wartość MQHCONN powiązaną z **odwołaniem do połączenia**. Wartość ta wynosi zero, jeśli nie ma **odwołań do połączenia** lub jeśli menedżer nie jest połączony.

ImqBoolean inquire (const MQLONG *int-attr*, MQLONG & *wartość*);

Zwraca wartość całkowitą, której indeks jest wartością MQIA_*. W przypadku błędu wartość jest ustawiana na wartość MQIAV_UNDEFINED.

ImqBoolean inquire (const MQLONG *char-attr*, char * & *buffer*, const size_t *długość*);

Zwraca łańcuch znaków, którego indeks jest wartością MQCA_*.

Uwaga: Obie te metody zwracają tylko jedną wartość atrybutu. Jeśli *obraz stanu* jest wymagany z więcej niż jednej wartości, gdzie wartości są spójne ze sobą przez chwilę, program IBM MQ C++ nie udostępnia tej funkcji, a użytkownik musi użyć wywołania MQINQ z odpowiednimi parametrami.

virtual void openInformationDisperse ();

Natychmiast po wywołaniu MQOPEN rozpraszaj informacje z sekcji zmiennej struktury danych MQOD.

virtual ImqBoolean openInformationPrepare ();

Przygotowuje informacje dla sekcji zmiennej struktury danych MQOD bezpośrednio przed wywołaniem wywołania MQOPEN i zwraca wartość PRAWDA, jeśli jest to pomyślne.

ImqBoolean set (const MQLONG int-attr, const MQLONG wartość);

Ustawia atrybut liczby całkowitej IBM MQ .

ImqBoolean set (const MQLONG char-attr, const char * buffer, const size_t wymagana-długość_czasu);

Ustawia atrybut znaku IBM MQ .

void setNextManagedObject (const ImqObject * obiekt = 0);

Ustawia **następny obiekt zarządzany**.

Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik jest pewien, że nie będzie przerywać listy obiektów zarządzanych.

void setPreviousManagedObject (const ImqObject * obiekt = 0);

Ustawia **poprzedni obiekt zarządzany**.

Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik jest pewien, że nie będzie przerywać listy obiektów zarządzanych.

Dane obiektu (chronione)**MQHOBJ ohobj**

Uchwył obiektu IBM MQ (poprawny tylko wtedy, gdy **otwarty status** ma wartość PRAWDA).

MQOD omqod

Wbudowana struktura danych MQOD. Ilość pamięci przydzielonej dla tej struktury danych jest wymagana dla programu MQOD w wersji 2. Sprawdź numer wersji (*omqod.Version*) i uzyskaj dostęp do innych pól w następujący sposób:

MQOD_VERSION_1

Można uzyskać dostęp do wszystkich pozostałych pól w katalogu *omqod* .

MQOD_VERSION_2

Można uzyskać dostęp do wszystkich pozostałych pól w katalogu *omqod* .

MQOD_VERSION_3

omqod.pmqod jest wskaźnikiem do dynamicznie przydzielonego, większego, MQOD. Nie można uzyskać dostępu do innych pól w pliku *omqod* . Dostęp do wszystkich pól adresowanych przez produkt *omqod.pmqod* można uzyskać.

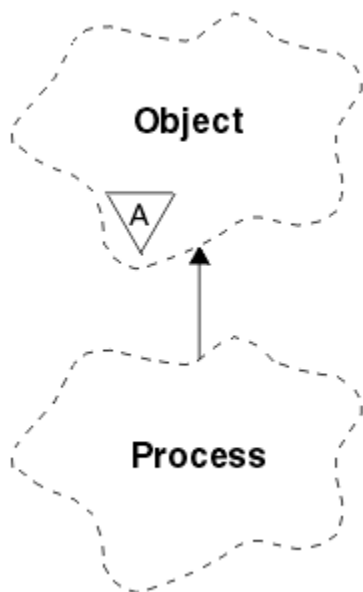
Uwaga: Wartość *omqod.pmqod.Version* może być mniejsza niż *omqod.Version*, co oznacza, że serwer IBM MQ MQI client ma więcej funkcji niż serwer IBM MQ .

Kody przyczyny

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (kody przyczyny z tabeli MQCLOSE)
- (kody przyczyny z tabeli MQCONN)
- (kody przyczyny z tabeli MQINQ)
- (kody przyczyny z komendy MQOPEN)
- (kody przyczyny z tabeli MQSET)

Klasa ImqProcess C++

Ta klasa hermetyzuje proces aplikacji (obiekt IBM MQ typu MQOT_PROCESS), który może być wyzwalany przez monitor wyzwalacza.



Rysunek 62. Klasa ImqProcess

- [“Atrybuty obiektu” na stronie 1989](#)
- [“Konstruktory” na stronie 1989](#)
- [“Metody obiektów \(publiczne\)” na stronie 1989](#)

Atrybuty obiektu

Identyfikator aplikacji

Tożsamość procesu aplikacji. Ten atrybut jest tylko do odczytu.

Typ aplikacji

Typ procesu aplikacji. Ten atrybut jest tylko do odczytu.

Dane środowiska

Informacje o środowisku dla procesu. Ten atrybut jest tylko do odczytu.

Dane użytkownika

Dane użytkownika dla procesu. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqProcess();

Konstruktor domyślny.

ImqProcess(const ImqProcess & proces);

Konstruktor kopiowania. Obiekt ImqObject **open status** ma wartość FALSE.

ImqProcess(const char * nazwa);

Ustawia nazwę ImqObject **nazwa**.

Metody obiektów (publiczne)

void operator = (const ImqProcess & proces);

Wykonuje zamknięcie, jeśli jest to konieczne, a następnie kopiuje dane instancji z *procesu*. Obiekt ImqObject **open status** będzie miał wartość FALSE.

ImqBoolean applicationId (ImqString & id);

Udostępnia kopię **identyfikatora aplikacji**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString applicationId ();

Zwraca **identyfikator aplikacji** bez wskazania ewentualnych błędów.

ImqBoolean applicationType (MQLONG & typ);

Udostępnia kopię **typu aplikacji**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG applicationType ();

Zwraca **typ aplikacji** bez wskazania ewentualnych błędów.

ImqBoolean environmentData (ImqString & data);

Udostępnia kopię **danych środowiska**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString environmentData ();

Zwraca **dane środowiska** bez wskazania ewentualnych błędów.

ImqBoolean userData (ImqString & dane);

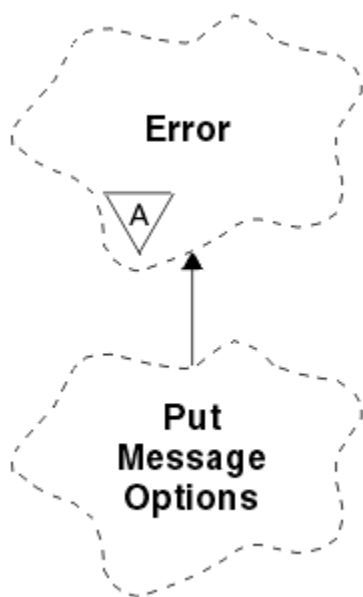
Udostępnia kopię **danych użytkownika**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString userData ();

Zwraca **dane użytkownika** bez wskazania ewentualnych błędów.

Klasa języka C++ ImqPutMessageOptions

Ta klasa hermetyzuje strukturę danych MQPMO.



Rysunek 63. Klasa ImqPutMessageOptions

- [“Atrybuty obiektu” na stronie 1990](#)
- [“Konstruktor” na stronie 1991](#)
- [“Metody obiektów \(publiczne\)” na stronie 1991](#)
- [“Dane obiektu \(chronione\)” na stronie 1992](#)
- [“Kody przyczyny” na stronie 1992](#)

Atrybuty obiektu

odwołanie do kontekstu

Kolejka ImqQueue , która udostępnia kontekst dla komunikatów. Początkowo nie ma żadnego odniesienia.

Opcje

Opcje umieszczania komunikatów. Wartością początkową jest MQPMO_NONE. Możliwe są następujące wartości dodatkowe:

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT
- MQPMO_NEW_MSG_ID
- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF QUIESCING

pola rekordu

Flagi sterujące włączeniem rekordów komunikatów umieszczanych w momencie umieszczania komunikatu. Wartością początkową jest MQPMRF_NONE. Możliwe są następujące wartości dodatkowe:

- MQPMRF_MSG_ID,
- MQPMRF_CORREL_ID
- Identyfikator MQPMRF_GROUP_ID
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN,

ImqMessage-atrybuty śledzenia są pobierane z obiektu dla dowolnego pola, które jest określone. Atrybuty ImqMessageTracker są pobierane z obiektu ImqMessage dla dowolnego pola, które *nie* jest określone.

rozstrzygnięta nazwa menedżera kolejek

Nazwa docelowego menedżera kolejek określonego podczas operacji put. Początkowa wartość jest równa null. Ten atrybut jest tylko do odczytu.

rozstrzygnięta nazwa kolejki

Nazwa kolejki docelowej określona podczas operacji put. Początkowa wartość jest równa null. Ten atrybut jest tylko do odczytu.

uczestnictwo w punkcie synchronizacji

PRAWDA, gdy komunikaty są umieszczane w elemencie sterującym punktu synchronizacji.

Konstruktory

ImqPutMessageOptions();

Konstruktor domyślny.

ImqPutMessageOptions(const ImqPutMessageOptions & pmo);

Konstruktor kopiowania.

Metody obiektów (publiczne)

void operator = (const ImqPutMessageOptions & pmo);

Kopiuje dane instancji z *pmo*, zastępując istniejące dane instancji.

ImqQueue * contextReference () const ;

Zwraca **odwołanie do kontekstu**.

void setContextReference (const ImqQueue & kolejka);

Ustawia **odwołanie do kontekstu**.

void setContextReference (const ImqQueue * kolejka = 0);

Ustawia **odwołanie do kontekstu**.

MQLONG opcje () const ;

Zwraca **opcje**.

void setOptions (const MQLONG opcje);

Ustawia wartość **options**, w tym wartość **syncpoint participation** .

MQLONG recordFields () const ;

Zwraca **pola rekordu**.

void setRecordFields (const MQLONG pola);

Ustawia **pola rekordu**.

ImqString resolvedQueueManagerName () const ;

Zwraca kopię **nazwy rozstrzygniętego menedżera kolejek**.

ImqString resolvedQueueNazwa () const ;

Zwraca kopię **przetłumaczanej nazwy kolejki**.

ImqBoolean syncPointUdział () const ;

Zwraca wartość **syncpoint participation** , która jest równa TRUE, jeśli **options** to MQPMO_SYNCPOINT.

void setSyncPointParticipation (const ImqBoolean sync);

Ustawia wartość parametru **syncpoint participation** . Jeśli parametr *sync* ma wartość TRUE, **opcje** są zmieniane w celu uwzględnienia MQPMO_SYNCPOINT i z wykluczeniem MQPMO_NO_SYNCPOINT. Jeśli parametr *sync* ma wartość FALSE, **opcje** są zmieniane w celu uwzględnienia MQPMO_NO_SYNCPOINT i do wykluczenia MQPMO_SYNCPOINT.

Dane obiektu (chronione)

MQPMO omqpmo

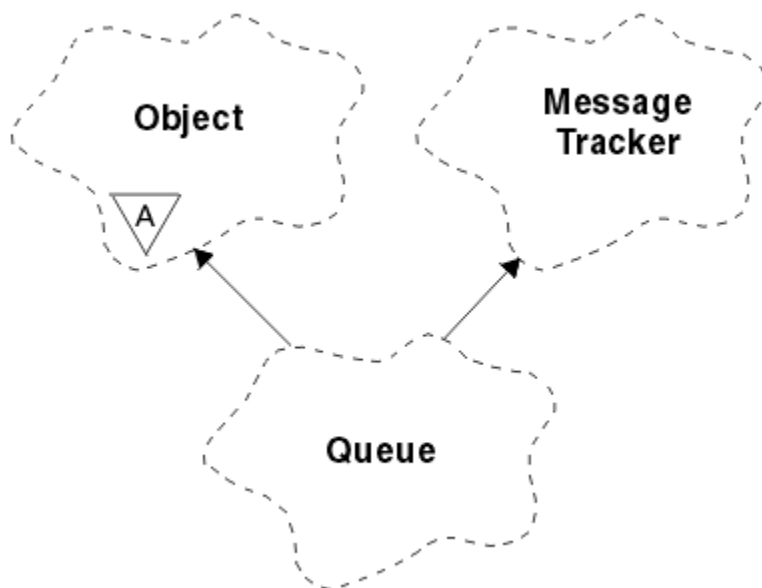
Struktura danych MQPMO.

Kody przyczyny

- MQRC_STORAGE_NOT_AVAILABLE

Klasa ImqQueue C++

Ta klasa hermetyzuje kolejkę komunikatów (obiekt IBM MQ typu MQOT_Q).



Rysunek 64. Klasa *ImqQueue*

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [Tabela 223 na stronie 1930](#).

- [“Atrybuty obiektu” na stronie 1993](#)
- [“Konstruktory” na stronie 1996](#)
- [“Metody obiektów \(publiczne\)” na stronie 1996](#)
- [“Metody obiektów \(chronione\)” na stronie 2003](#)
- [“Kody przyczyny” na stronie 2003](#)

Atrybuty obiektu

nazwa kolejki wycofanych komunikatów

Nadmierna nazwa kolejki wycofanych komunikatów. Ten atrybut jest tylko do odczytu.

Próg wycofania

Próg wycofania. Ten atrybut jest tylko do odczytu.

podstawowa nazwa kolejki

Nazwa kolejki, do której jest tłumaczona alias. Ten atrybut jest tylko do odczytu.

nazwa klastra

Nazwa klastra. Ten atrybut jest tylko do odczytu.

Nazwa listy nazw klastrów

Nazwa listy nazw klastrów. Ten atrybut jest tylko do odczytu.

Klasyfikacja obciążenia klastrów

Stopień obciążenia klastra. Ten atrybut jest tylko do odczytu.

Priorytet obciążenia klastrów

Priorytet obciążenia klastra. Ten atrybut jest tylko do odczytu.

Kolejka użycia obciążenia klastra

Wartość kolejki użycia obciążenia klastra. Ten atrybut jest tylko do odczytu.

Data utworzenia

Dane tworzenia kolejki. Ten atrybut jest tylko do odczytu.

Czas utworzenia

Czas utworzenia kolejki. Ten atrybut jest tylko do odczytu.

Bieżące zapełnienie

Liczba komunikatów w kolejce. Ten atrybut jest tylko do odczytu.

powiązanie domyślne

Powiązanie domyślne. Ten atrybut jest tylko do odczytu.

Domyślna opcja otwarcia wejścia

Domyślna opcja open-for-input. Ten atrybut jest tylko do odczytu.

Trwałość domyślna

Domyślna trwałość komunikatu. Ten atrybut jest tylko do odczytu.

Domyślny priorytet

Domyślny priorytet komunikatu. Ten atrybut jest tylko do odczytu.

Typ definicji

Typ definicji kolejki. Ten atrybut jest tylko do odczytu.

high event

Atrybut elementu sterującego dla zdarzeń wysokiego zapełnienia kolejki. Ten atrybut jest tylko do odczytu.

głębokość górnego limitu

Górny limit głębokości kolejki. Ten atrybut jest tylko do odczytu.

zdarzenie o niskiej głębokości

Atrybut elementu sterującego dla zdarzeń o niskiej głębokości kolejki. Ten atrybut jest tylko do odczytu.

dolny limit głębokości

Niski limit głębokości kolejki. Ten atrybut jest tylko do odczytu.

maksymalna głębokość zdarzenia

Atrybut elementu sterującego dla zdarzeń maksymalnej głębokości kolejki. Ten atrybut jest tylko do odczytu.

odwołanie do listy dystrybucji

Opcjonalne odwołanie do listy ImqDistribution, które może być używane do dystrybuowania komunikatów do więcej niż jednej kolejki, w tym do tej listy. Początkowa wartość jest równa null.

Uwaga: Po otwarciu obiektu ImqQueue każdy otwarty obiekt listy ImqDistribution, do którego odwołuje się ten obiekt, jest automatycznie zamykany.

listy dystrybucyjne

Możliwość obsługi kolejki transmisji w celu obsługi list dystrybucyjnych. Ten atrybut jest tylko do odczytu.

nazwa kolejki dynamicznej

Nazwa kolejki dynamicznej. Wartością początkową jest AMQ.* dla wszystkich platform Windows, UNIX i Linux.

Zapisane wycofane komunikaty

Określa, czy liczba wycofań ma być utwardzona. Ten atrybut jest tylko do odczytu.

Typ indeksu

Typ indeksu. Ten atrybut jest tylko do odczytu.

inhibit get (pobieranie)

Określa, czy operacje pobierania są dozwolone. Wartość początkowa jest zależna od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku aliasu lub kolejki lokalnej.

umieszczanie zablokowanej strony

Określa, czy operacje put są dozwolone. Wartość początkowa jest zależna od definicji kolejki.

Nazwa kolejki inicjacji

Nazwa kolejki inicjuj. Ten atrybut jest tylko do odczytu.

Zapełnienie maksymalne

Maksymalna liczba komunikatów dozwolonych w kolejce. Ten atrybut jest tylko do odczytu.

Maksymalna długość komunikatu

Maksymalna długość dowolnego komunikatu w tej kolejce, która może być mniejsza niż wartość maksymalna dla dowolnej kolejki zarządzanej przez powiązany menedżer kolejek. Ten atrybut jest tylko do odczytu.

Kolejność dostarczania komunikatów

Określa, czy priorytet komunikatu jest istotny. Ten atrybut jest tylko do odczytu.

Następna kolejka rozproszona

Następny obiekt tej klasy, w żadnym konkretnym porządku, o tej samej **odwołaniu do listy dystrybucyjnej**, co ten obiekt. Wartością początkową jest zero.

Jeśli obiekt w łańcuchu zostanie usunięty, poprzedni obiekt i następny obiekt zostaną zaktualizowane w taki sposób, aby ich rozproszone połączenia kolejki nie wskazywały już na usunięty obiekt.

nietrwała klasa komunikatu

Poziom niezawodności dla nietrwałych komunikatów umieszczonych w tej kolejce. Ten atrybut jest tylko do odczytu.

Liczba otwartych wejść

Liczba obiektów ImqQueue, które są otwarte na dane wejściowe. Ten atrybut jest tylko do odczytu.

Liczba otwartych wyjść

Liczba obiektów ImqQueue, które są otwarte na dane wyjściowe. Ten atrybut jest tylko do odczytu.

poprzednia kolejka rozproszona

Poprzedni obiekt tej klasy, w żadnym konkretnym porządku, o tej samej **odwołaniu do listy dystrybucyjnej** co ten obiekt. Wartością początkową jest zero.

Jeśli obiekt w łańcuchu zostanie usunięty, poprzedni obiekt i następny obiekt zostaną zaktualizowane w taki sposób, aby ich rozproszone połączenia kolejki nie wskazywały już na usunięty obiekt.

Nazwa procesu

Nazwa definicji procesu. Ten atrybut jest tylko do odczytu.

Rozliczanie kolejek

Poziom informacji rozliczeniowych dla kolejek. Ten atrybut jest tylko do odczytu.

Nazwa menedżera kolejek

Nazwa menedżera kolejek (ewentualnie zdalnego), w którym znajduje się kolejka. Nie należy mylić menedżera kolejek określonego w tym miejscu za pomocą ImqObject **odwołanie do połączenia**, który odwołuje się do (lokalnego) menedżera kolejek udostępniających połączenie. Początkowa wartość jest równa null.

Monitorowanie kolejek

Poziom gromadzenia danych monitorowania dla kolejki. Ten atrybut jest tylko do odczytu.

Statystyka kolejek

Poziom danych statystycznych dla kolejki. Ten atrybut jest tylko do odczytu.

Typ kolejki

Typ kolejki. Ten atrybut jest tylko do odczytu.

Nazwa zdalnego menedżera kolejek

Nazwa zdalnego menedżera kolejek. Ten atrybut jest tylko do odczytu.

Nazwa zdalnej kolejki

Nazwa kolejki zdalnej, która jest znana w zdalnym menedżerze kolejek. Ten atrybut jest tylko do odczytu.

rozstrzygnięta nazwa menedżera kolejek

Rozstrzygnięta nazwa menedżera kolejek. Ten atrybut jest tylko do odczytu.

rozstrzygnięta nazwa kolejki

Rozstrzygnięta nazwa kolejki. Ten atrybut jest tylko do odczytu.

Interwał przechowywania

Interwał czasu przechowywania kolejki. Ten atrybut jest tylko do odczytu.

Scope

Zasięg definicji kolejki. Ten atrybut jest tylko do odczytu.

interwał usług

Przedział czasu usługi. Ten atrybut jest tylko do odczytu.

zdarzenie interwału usług

Atrybut elementu sterującego dla zdarzeń odstępu czasu usługi. Ten atrybut jest tylko do odczytu.

Możliwość współużytkowania

Określa, czy kolejka może być współużytkowana. Ten atrybut jest tylko do odczytu.

klasa pamięci masowej

Klasa pamięci. Ten atrybut jest tylko do odczytu.

Nazwa kolejki transmisji

Nazwa kolejki przesyłania. Ten atrybut jest tylko do odczytu.

Kontrola wyzwacza

Sterowanie wyzwaczem. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku kolejki lokalnej.

Dane wyzwacza

Dane wyzwacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku kolejki lokalnej.

Wyzwalacz uruchamiany zapętnieniem

Wyzwalacz uruchamiany zapętnieniem. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku kolejki lokalnej.

Priorytet komunikatu wyzwacza

Priorytet komunikatu progowego dla wyzwacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku kolejki lokalnej.

typ wyzwacza

Typ wyzwacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko w przypadku kolejki lokalnej.

Składnia

Użycie. Ten atrybut jest tylko do odczytu.

Konstruktory**ImqQueue();**

Konstruktor domyślny.

ImqQueue(const ImqQueue & kolejka);

Konstruktor kopiowania. Obiekt ImqObject **open status** będzie miał wartość FALSE.

ImqQueue(const char * nazwa);

Ustawia nazwę ImqObject **nazwa**.

Metody obiektów (publiczne)**void operator = (const ImqQueue & kolejka);**

Wykonuje zamknięcie, jeśli jest to konieczne, a następnie kopiuje dane instancji z *kolejki*. Obiekt ImqObject **open status** będzie miał wartość FALSE.

ImqBoolean backoutRequeueNazwa (ImqString & nazwa);

Udostępnia kopię **nazwy wycofanych komunikatów wycofanych**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString backoutRequeueNazwa ();

Zwraca **nazwę kolejki wycofanych kopii zapasowych** bez wskazania ewentualnych błędów.

ImqBoolean backoutThreshold (MQLONG & próg);

Udostępnia kopię **progu wycofania**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG backoutThreshold ();

Zwraca wartość **progu wycofania** bez wskazania możliwych błędów.

ImqBoolean baseQueueNazwa (ImqString & nazwa);

Udostępnia kopię **nazwy kolejki podstawowej**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString baseQueueNazwa ();

Zwraca **nazwę kolejki podstawowej** bez wskazania ewentualnych błędów.

ImqBoolean clusterName(ImqString & nazwa);

Udostępnia kopię **nazwy klastra**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString clusterName ();

Zwraca **nazwę klastra** bez wskazania ewentualnych błędów.

ImqBoolean clusterNamelistName (ImqString & nazwa);

Udostępnia kopię **nazwy listy nazw klastra**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString clusterNamelistName ();

Zwraca **nazwę listy nazw klastra** bez podawania informacji o błędach.

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);

Udostępnia kopię wartości priorytetu obciążenia klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkLoadPriority ();

Zwraca wartość priorytetu obciążenia klastra bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkLoadRank (MQLONG & rank);

Udostępnia kopię wartości oceny obciążenia klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkLoadRank ();

Zwraca wartość oceny obciążenia klastra bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);

Udostępnia kopię wartości kolejki użycia obciążenia klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkLoadUseQ ();

Zwraca wartość kolejki wykorzystania obciążenia klastra bez wskazania ewentualnych błędów.

ImqBoolean creationDate (ImqString & data);

Udostępnia kopię **daty utworzenia**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString creationDate ();

Zwraca **datę utworzenia** bez wskazania ewentualnych błędów.

ImqBoolean creationTime (ImqString & czas);

Udostępnia kopię **czasu utworzenia**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString creationTime ();

Zwraca **czas utworzenia** bez wskazania ewentualnych błędów.

ImqBoolean currentDepth (MQLONG & głębokość);

Udostępnia kopię **bieżącej głębokości**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG currentDepth ();

Zwraca **bieżące zapętnienie** bez wskazania ewentualnych błędów.

ImqBoolean defaultInputOpenOption (MQLONG & opcja);

Udostępnia kopię **domyślnej opcji otwierania danych wejściowych**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG defaultInputOpenOption ();

Zwraca **domyślną otwartą opcję wejścia** bez wskazania możliwych błędów.

ImqBoolean defaultPersistence (MQLONG & trwałość);

Udostępnia kopię **domyślnej trwałości**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG defaultPersistence ();

Powoduje zwrócenie wartości **default persistence** bez wskazania ewentualnych błędów.

ImqBoolean defaultPriority (MQLONG & priorytet);

Udostępnia kopię **domyślnego priorytetu**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG defaultPriority ();

Zwraca **domyślny priorytet** bez wskazania ewentualnych błędów.

ImqBoolean defaultBind (MQLONG & bind);

Udostępnia kopię **domyślnego powiązania**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG defaultBind ();

Zwraca wartość **default bind** bez wskazania ewentualnych błędów.

ImqBoolean definitionType (MQLONG & typ);

Udostępnia kopię **typu definicji**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG definitionType ();

Zwraca **typ definicji** bez wskazania ewentualnych błędów.

ImqBoolean depthHighZdarzenie (MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia wysokiego zapętnienia**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG depthHighEvent ();

Zwraca stan włączenia **zdarzenia wysokiego zapętnienia** bez wskazania ewentualnych błędów.

ImqBoolean depthHighLimit (MQLONG & limit);

Udostępnia kopię **górnego limitu głębokości**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG depthHighLimit ();

Zwraca wartość parametru **głębokość wysokiego poziomu** bez wskazania ewentualnych błędów.

ImqBoolean depthLowZdarzenie (MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia niskiego zapętnienia**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG depthLowzdarzenie ();

Zwraca stan włączenia **zdarzenia niskiego poziomu głębokości** bez wskazania ewentualnych błędów.

ImqBoolean depthLowLimit (MQLONG & limit);

Udostępnia kopię **limitu niskiego poziomu głębokości**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG depthLowLimit ();

Zwraca wartość parametru **low low limit** bez wskazania ewentualnych błędów.

ImqBoolean depthMaximumZdarzenie (MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **maksymalnego zdarzenia głębokości**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG depthMaximumZdarzenie ();

Zwraca stan włączenia **zdarzenia maksymalnego zapętnienia** bez wskazania ewentualnych błędów.

Lista ImqDistribution* distributionListReference () const ;

Zwraca **odwołanie do listy dystrybucyjnej**.

void setDistributionListReference (ImqDistributionLista & lista);

Ustawia **odwołanie do listy dystrybucyjnej**.

void setDistributionListReference (ImqDistributionList * list = 0);

Ustawia **odwołanie do listy dystrybucyjnej**.

ImqBoolean distributionLists (MQLONG & obsługa);

Udostępnia kopię wartości **list dystrybucyjnych** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG distributionLists ();

Zwraca wartość **list dystrybucyjnych** bez wskazania ewentualnych błędów.

ImqBoolean setDistributionLists (const MQLONG obsługa);

Ustawia wartość **list dystrybucyjnych** . Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString dynamicQueueNazwa () const ;

Zwraca kopię **nazwy kolejki dynamicznej**.

ImqBoolean setDynamicQueueName (const char * nazwa);

Ustawia **nazwę kolejki dynamicznej**. Nazwa kolejki dynamicznej może być ustawiona tylko wtedy, gdy ImqObject **open status** ma wartość FALSE. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean get (ImqMessage & msg, ImqGetMessageOptions & opcje);

Pobiera komunikat z kolejki przy użyciu określonej opcji *opcje*. Wywołuje metodę ImqObject **openFor** , jeśli jest to konieczne, aby upewnić się, że ImqObject **open options** zawiera jedną z wartości MQOO_INPUT_ * lub wartość MQOO_BROWSE, w zależności od *opcji*. Jeśli obiekt *komunikat*

ma `ImqCache` **bufor automatyczny**, bufor rośnie w celu uwzględnienia wszystkich pobranych komunikatów. Metoda **`clearMessage`** jest wywoływana względem obiektu `msg` przed pobraniem.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Uwaga: Wynikiem wywołania metody jest FALSE, jeśli `ImqObject` **kod przyczyny** to `MQRC_TRUNCATED_MSG_FAILED`, nawet jeśli ten **kod przyczyny** jest klasyfikowany jako ostrzeżenie. Jeśli obcięty komunikat jest akceptowany, wartość `ImqCache` **długość komunikatu** odzwierciedla obciętą długość. W obu tych zdarzeniach wartość `ImqMessage` **łączna długość komunikatu** wskazuje liczbę bajtów, które były dostępne.

`ImqBoolean get ( ImqMessage & komunikat );`

Podobnie jak w przypadku poprzedniej metody, z tą różnicą, że używane są domyślne opcje pobierania komunikatów.

`ImqBoolean get ( ImqMessage & msg, ImqGetMessageOptions & options, const size_t buffer-size );`

Podobnie jak w przypadku poprzednich dwóch metod, z tą różnicą, że wskazana jest przestawianie wielkość-buforu . Jeśli obiekt *komunikat* korzysta z `ImqCache` **bufor automatyczny**, metoda **`resizeBuffer`** jest wywoływana w obiekcie *komunikat* przed pobraniem komunikatów, a bufor nie powiększa się, aby pomieścić większą wiadomość.

`ImqBoolean get ( ImqMessage & msg, const size_t wielkość_buforu );`

Podobnie jak w przypadku poprzedniej metody, z tą różnicą, że używane są domyślne opcje pobierania komunikatów.

`ImqBoolean hardenGetBackout ( MQLONG & harden );`

Udostępnia kopię wartości **`harden get backout`** (`harden get backout`). Zwraca wartość PRAWDA, jeśli powiodła się.

`MQLONG hardenGetBackout ();`

Zwraca wartość **`harden get backout`** bez wskazania ewentualnych błędów.

`ImqBoolean indexType(MQLONG & typ );`

Udostępnia kopię **typu indeksu**. Zwraca wartość PRAWDA, jeśli powiodła się.

`MQLONG indexType();`

Zwraca **typ indeksu** bez wskazania ewentualnych błędów.

`ImqBoolean inhibitGet ( MQLONG & inhibit );`

Udostępnia kopię wartości **`inhibit get`** . Zwraca wartość PRAWDA, jeśli powiodła się.

`MQLONG inhibitGet ();`

Zwraca wartość **`inhibit get`** bez wskazania ewentualnych błędów.

`ImqBoolean setInhibitGet ( const MQLONG inhibit );`

Ustawia wartość parametru **`inhibit get`** . Zwraca wartość PRAWDA, jeśli powiodła się.

`ImqBoolean inhibitPut ( MQLONG & inhibit );`

Udostępnia kopię wartości **`zablokuj wstawione`** . Zwraca wartość PRAWDA, jeśli powiodła się.

`MQLONG inhibitPut ();`

Zwraca wartość **`inhibit put`** bez wskazania ewentualnych błędów.

`ImqBoolean setInhibitPut ( const MQLONG inhibit );`

Ustawia wartość parametru **`inhibit put`** . Zwraca wartość PRAWDA, jeśli powiodła się.

`ImqBoolean initiationQueueNazwa ( ImqString & nazwa );`

Udostępnia kopię **`nazwy kolejki inicjuj`**. Zwraca wartość PRAWDA, jeśli powiodła się.

`ImqString initiationQueueNazwa ();`

Zwraca **`nazwę kolejki inicjuj`**. bez wskazania ewentualnych błędów.

`ImqBoolean maximumDepth ( MQLONG & głębokość );`

Udostępnia kopię **`maksymalnej głębokości`**. Zwraca wartość PRAWDA, jeśli powiodła się.

`MQLONG maximumDepth ();`

Zwraca **`maksymalną głębokość`** bez wskazania ewentualnych błędów.

`ImqBoolean maximumMessageLength ( MQLONG & długość );`

Udostępnia kopię **`maksymalnej długości komunikatu`**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumMessageLength ();

Zwraca **maksymalną długość komunikatu** bez wskazania ewentualnych błędów.

ImqBoolean messageDeliverySekuencja (MQLONG & sekuencja);

Udostępnia kopię **sekuencji dostarczania komunikatów**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG messageDeliveryKolejność ();

Zwraca wartość **sekuencji dostarczania komunikatów** bez wskazania ewentualnych błędów.

ImqQueue * nextDistributedQueue () const ;

Zwraca **następną kolejkę rozproszoną**.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);

Udostępnia kopię wartości klasy komunikatu nietrwałego. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG nonPersistentMessageClass ();

Zwraca wartość klasy nietrwałej komunikatu bez wskazania ewentualnych błędów.

ImqBoolean openInputCount (MQLONG & liczba);

Udostępnia kopię **otwartego licznika danych wejściowych**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG openInputCount ();

Zwraca **liczbę otwartych wejść** bez wskazania możliwych błędów.

ImqBoolean openOutputCount (MQLONG & liczba);

Udostępnia kopię **otwartego licznika danych wyjściowych**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG openOutputCount ();

Zwraca **liczbę otwartych danych wyjściowych** bez wskazania możliwych błędów.

ImqQueue * previousDistributedQueue () const ;

Zwraca **poprzednią kolejkę rozproszoną**.

ImqBoolean processName (ImqString & nazwa);

Udostępnia kopię **nazwy procesu**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString processName ();

Zwraca **nazwę procesu** bez wskazania ewentualnych błędów.

ImqBoolean put (ImqMessage & komunikat);

Umieszcza komunikat w kolejce przy użyciu domyślnych opcji umieszczania komunikatów. Używa metody ImqObject **openFor** , jeśli jest to konieczne, aby upewnić się, że ImqObject **open options** to MQOO_OUTPUT.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean put (ImqMessage & msg, ImqPutMessageOptions & pmo);

Umieszcza komunikat w kolejce przy użyciu podanego *pmo*. Metoda ImqObject **openFor** jest używana w razie potrzeby w celu zapewnienia, że ImqObject **open options** to MQOO_OUTPUT, oraz (jeśli *pmo options* zawiera dowolny z wartości MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT), które odpowiadają wartości MQOO_*_CONTEXT.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Uwaga: Jeśli *pmo* zawiera **odwołanie do kontekstu**, przywoływany obiekt jest otwierany, jeśli jest to konieczne, w celu udostępnienia kontekstu.

ImqBoolean queueAccounting (MQLONG & acctq);

Udostępnia kopię wartości rozliczania kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueAccounting ();

Zwraca wartość księgową kolejki bez wskazania ewentualnych błędów.

ImqString queueManagerNazwa () const ;

Zwraca **nazwę menedżera kolejek**.

ImqBoolean setQueueManagerName (const char * *nazwa*);
 Ustawia **nazwę menedżera kolejek**. **Nazwa menedżera kolejek** może być ustawiona tylko wtedy, gdy ImqObject **open status** ma wartość FALSE. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean queueMonitoring (MQLONG & monq);
 Udostępnia kopię wartości monitorowania kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueMonitoring ();
 Zwraca wartość monitorowania kolejki bez wskazania ewentualnych błędów.

ImqBoolean queueStatistics (MQLONG & statq);
 Udostępnia kopię wartości statystyki kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueStatistics ();
 Zwraca wartość statystyki kolejki bez wskazania ewentualnych błędów.

ImqBoolean queueType (MQLONG & typ);
 Udostępnia kopię wartości **typ kolejki** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueType ();
 Zwraca **typ kolejki** bez wskazania ewentualnych błędów.

ImqBoolean remoteQueueManagerName (ImqString & nazwa);
 Udostępnia kopię **nazwy zdalnego menedżera kolejek**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString remoteQueueManagerName ();
 Zwraca **nazwę zdalnego menedżera kolejek** bez wskazania ewentualnych błędów.

ImqBoolean remoteQueueNazwa (ImqString & nazwa);
 Udostępnia kopię **nazwy kolejki zdalnej**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString remoteQueueNazwa ();
 Zwraca **nazwę kolejki zdalnej** bez wskazania ewentualnych błędów.

ImqBoolean resolvedQueueManagerName(ImqString & nazwa);
 Udostępnia kopię **nazwy rozstrzygniętego menedżera kolejek**. Zwraca wartość PRAWDA, jeśli powiodła się.

Uwaga: Ta metoda kończy się niepowodzeniem, chyba że parametr MQOO_RESOLVE_NAMES znajduje się wśród opcji ImqObject **open options**.

ImqString resolvedQueueManagerName() ;
 Zwraca **nazwę rozstrzygniętego menedżera kolejek**, bez wskazania ewentualnych błędów.

ImqBoolean resolvedQueueNazwa (ImqString & nazwa);
 Udostępnia kopię **nazwy rozstrzygniętej kolejki**. Zwraca wartość PRAWDA, jeśli powiodła się.

Uwaga: Ta metoda kończy się niepowodzeniem, chyba że parametr MQOO_RESOLVE_NAMES znajduje się wśród opcji ImqObject **open options**.

ImqString resolvedQueueName ();
 Zwraca **rozstrzygniętą nazwę kolejki** bez wskazania ewentualnych błędów.

ImqBoolean retentionInterval (MQLONG & interwał);
 Udostępnia kopię **przedziału czasu przechowywania**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG retentionInterval ();
 Zwraca **przedział czasu przechowywania** bez wskazania możliwych błędów.

ImqBoolean zasięg (MQLONG & zasięg);
 Udostępnia kopię **zasięgu**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG zasięg ();
 Zwraca **zasięg** bez wskazania możliwych błędów.

ImqBoolean serviceInterval (MQLONG & interwał);
 Udostępnia kopię **przedziału czasu usługi**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG serviceInterval ();
 Zwraca **przedział czasu usługi** bez wskazania ewentualnych błędów.

ImqBoolean serviceIntervalZdarzenie (MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia odstępu czasu usługi**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG serviceIntervalZdarzenie ();

Zwraca stan włączenia **zdarzenia odstępu czasu usługi** bez wskazania ewentualnych błędów.

ImqBoolean shareability (MQLONG & shareability);

Udostępnia kopię wartości **shareability** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG współużytkowność ();

Zwraca wartość **shareability** bez wskazania ewentualnych błędów.

ImqBoolean storageClass(ImqString & klasa);

Udostępnia kopię **klasy pamięci masowej**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString storageClass();

Zwraca **klasę pamięci masowej** bez wskazania ewentualnych błędów.

ImqBoolean transmissionQueueNazwa (ImqString & nazwa);

Udostępnia kopię **nazwy kolejki transmisji**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString transmissionQueueName ();

Zwraca **nazwę kolejki transmisji** bez wskazania ewentualnych błędów.

ImqBoolean triggerControl (MQLONG & element sterujący);

Udostępnia kopię wartości **control control** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG triggerControl ();

Zwraca wartość parametru **trigger control** bez wskazania ewentualnych błędów.

ImqBoolean setTriggerControl (const MQLONG control);

Ustawia wartość parametru **trigger control** . Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean triggerData (ImqString & data);

Udostępnia kopię **danych wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString triggerData ();

Zwraca kopię **danych wyzwalacza** bez wskazania ewentualnych błędów.

ImqBoolean setTriggerData (const char * dane);

Ustawia **dane wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean triggerDepth (MQLONG & głębokość);

Udostępnia kopię **głębokości wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG triggerDepth ();

Zwraca **głębokość wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerDepth (const MQLONG głębokość);

Ustawia **głębokość wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean triggerMessagePriorytet (MQLONG & priorytet);

Udostępnia kopię **priorytetu komunikatu wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG triggerMessagePriorytet ();

Zwraca **priorytet komunikatu wyzwalacza** bez wskazania ewentualnych błędów.

ImqBoolean setTriggerMessagePriority (const MQLONG priorytet);

Ustawia **priorytet komunikatu wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean triggerType (MQLONG & typ);

Udostępnia kopię **typu wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG triggerType ();

Zwraca **typ wyzwalacza** bez wskazania ewentualnych błędów.

ImqBoolean setTriggerType (const MQLONG typ);

Ustawia **typ wyzwalacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean składnia (MQLONG & składnia);

Udostępnia kopię wartości **użycia** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG składnia ();

Zwraca wartość **usage** bez wskazania ewentualnych błędów.

Metody obiektów (chronione)

void setNextDistributedQueue (ImqQueue * kolejka = 0);

Ustawia **następną kolejkę rozproszoną**.

Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik jest pewien, że nie ma podziału listy kolejek rozproszonych.

void setPreviousDistributedQueue (ImqQueue * kolejka = 0);

Ustawia **poprzednią kolejkę rozproszoną**.

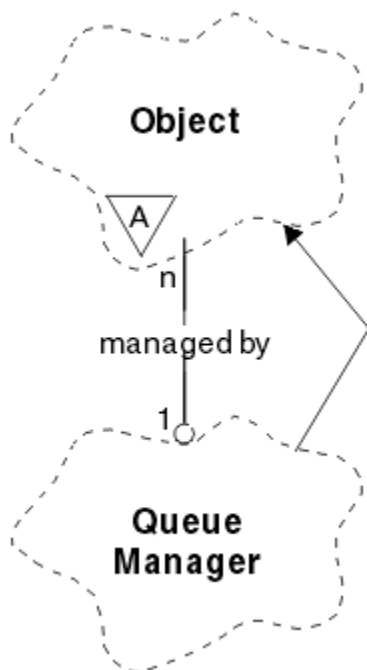
Uwaga: Ta funkcja jest używana tylko wtedy, gdy użytkownik jest pewien, że nie ma podziału listy kolejek rozproszonych.

Kody przyczyny

- 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
- (kody przyczyny z komendy MQGET)
- (kody przyczyny z MQPUT)

Klasa C++ programu ImqQueueManager

Ta klasa hermetyzuje menedżera kolejek (obiekt IBM MQ typu MQOT_Q_MGR).



Rysunek 65. Klasa menedżera ImqQueue

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“Odwołanie do menedżera ImqQueueManager”](#) na stronie 1932. Nie wszystkie wymienione metody mają zastosowanie do wszystkich platform. Więcej szczegółowych informacji na ten temat zawiera sekcja [ALTER QMGR](#).

- [“Atrybuty klasy”](#) na stronie 2004
- [“Atrybuty obiektu”](#) na stronie 2005
- [“Konstruktory”](#) na stronie 2010
- [“Destruktry”](#) na stronie 2010
- [“Metody klasy \(publiczne\)”](#) na stronie 2010
- [“Metody obiektów \(publiczne\)”](#) na stronie 2010
- [“Metody obiektów \(chronione\)”](#) na stronie 2019
- [“Dane obiektu \(chronione\)”](#) na stronie 2019
- [“Kody przyczyny”](#) na stronie 2019

Atrybuty klasy

zachowanie

Kontroluje zachowanie niejawnego połączenia i rozłączenia.

IMQ_EXPL_DISC_BACKOUT (0L)

Jawne wywołanie metody **disconnect** oznacza, że jest on wycofany. Ten atrybut wyklucza się wzajemnie z parametrem IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)

Jawne wywołanie metody **disconnect** oznacza zatwierdzenie (wartość domyślna). Ten atrybut wyklucza się wzajemnie z wartością IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)

Połączenie niejawne jest dozwolone (wartość domyślna).

IMQ_IMPL_DISC_BACKOUT (0L)

Niejawne wywołanie metody **disconnect**, które może wystąpić podczas destrukcji obiektu, oznacza cofanie. Ten atrybut wyklucza się wzajemnie z parametrem IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)

Niejawne wywołanie metody **disconnect**, które może wystąpić podczas destrukcji obiektu, oznacza zatwierdzenie (wartość domyślna). Ten atrybut wyklucza się wzajemnie z wartością IMQ_IMPL_DISC_BACKOUT.

W wersji IBM MQ V7.0 i nowszych, aplikacje C++, które korzystają z połączenia niejawnego, muszą określić IMQ_IMPL_CONN wraz z innymi opcjami dostarczonym w metodzie `setBehavior()` na obiekcie klasy `ImqQueueManager`. Jeśli aplikacja nie używa metody `setBehavior()` do jawnego ustawiania opcji zachowania, na przykład:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Ta zmiana nie wpływa na użytkownika, ponieważ wartość MQ_IMPL_CONN jest włączona domyślnie.

Jeśli aplikacja jawnie ustawi opcje zachowania, na przykład:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Aby umożliwić aplikacji zakończenie niejawnego połączenia, należy dołączyć IMQ_IMPL_CONN do metody `setBehavior()` w następujący sposób:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```


Atrybuty obiektu

nadpisywanie połączeń rozliczania

Allows applications to override the setting of the MQI accounting and queue accounting values. This attribute is read-only.

Interwał rozliczania

Jak długo przed zapisami pośrednich zapisów księgowych (w sekundach). Ten atrybut jest tylko do odczytu.

Zapis aktywności

Steruje generowaniem raportów działania. Ten atrybut jest tylko do odczytu.

Sprawdzenie dołączenia nowego MCA

Elementy sprawdzane w celu określenia, czy agent MCA ma zostać adoptowane po wykryciu nowego kanału danych przychodzących o tej samej nazwie co agent MCA, który już jest aktywny. Ten atrybut jest tylko do odczytu.

Typ dołączenia nowego MCA

Określa, czy osierocona instancja agenta MCA danego typu kanału powinna zostać zrestartowana automatycznie, gdy zostanie wykryte nowe żądanie kanału przychodzącego zgodne z nowymi parametrami sprawdzania mca. Ten atrybut jest tylko do odczytu.

Typ uwierzytelniania

Wskazuje typ uwierzytelniania, który jest wykonywany.

zdarzenie uprawnienia

Steruje zdarzeniami uprawnień. Ten atrybut jest tylko do odczytu.

opcje rozpoczęcia

Opcje, które mają zastosowanie do metody **begin** . Wartością początkową jest MQBO_NONE.

zdarzenie mostu

Określa, czy generowane są zdarzenia mostu IMS . Ten atrybut jest tylko do odczytu.

Automatyczna definicja kanału

Wartość automatycznego definiowania kanału. Ten atrybut jest tylko do odczytu.

zdarzenie automatycznego definiowania kanału

Wartość zdarzenia automatycznego definiowania kanału. Ten atrybut jest tylko do odczytu.

Wyjście automatycznej definicji kanału

Nazwa wyjścia automatycznej definicji kanału. Ten atrybut jest tylko do odczytu.

zdarzenie kanału

Określa, czy generowane są zdarzenia kanału. Ten atrybut jest tylko do odczytu.

Adaptery inicjatora kanału

Liczba podzadań adaptera, które mają być używane do przetwarzania wywołań IBM MQ . Ten atrybut jest tylko do odczytu.

Kontrola inicjatora kanału

Informacja o tym, czy inicjator kanału powinien być uruchamiany automatycznie podczas uruchamiania menedżera kolejek. Ten atrybut jest tylko do odczytu.

Programy rozsyłające inicjatora kanału

Liczba programów rozsyłających, które mają zostać użyte dla inicjatora kanału. Ten atrybut jest tylko do odczytu.

autostart śledzenia inicjatora kanału

Określa, czy śledzenie inicjatora kanału powinno być uruchamiane automatycznie, czy nie. Ten atrybut jest tylko do odczytu.

Wielkość tabeli śledzenia inicjatora kanału

Wielkość obszaru danych śledzenia inicjatora kanału (w MB). Ten atrybut jest tylko do odczytu.

Monitorowanie kanałów

Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kanałów. Ten atrybut jest tylko do odczytu.

odwołanie do kanału

Odwołanie do definicji kanału w celu użycia podczas nawiązywania połączenia z klientem. Podczas połączenia ten atrybut można ustawić na wartość NULL, ale nie można go zmienić na żadną inną wartość. Początkowa wartość jest równa null.

Statystyka kanałów

Steruje kolekcjonowaniem danych statystycznych dla kanałów. Ten atrybut jest tylko do odczytu.

zestaw znaków

Identyfikator kodowanego zestawu znaków (CCSID). Ten atrybut jest tylko do odczytu.

Monitorowanie nadawcy klastrów

Steruje gromadzeniem danych monitorowania w trybie z połączeniem dla automatycznie zdefiniowanych kanałów nadajnika klastrów. Ten atrybut jest tylko do odczytu.

Statystyka nadawcy klastrów

Steruje gromadzeniem danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastrów. Ten atrybut jest tylko do odczytu.

Dane obciążenia klastra

Dane wyjścia obciążenia klastra. Ten atrybut jest tylko do odczytu.

Wyjście obciążenia klastra

Nazwa wyjścia obciążenia klastra. Ten atrybut jest tylko do odczytu.

Długość obciążenia klastra

Długość obciążenia klastra. Ten atrybut jest tylko do odczytu.

obciążenie klastra mru

Obciążenie klastra ostatnio używane przez wartość kanałów. Ten atrybut jest tylko do odczytu.

Kolejka użycia obciążenia klastra

Wartość kolejki użycia obciążenia klastra. Ten atrybut jest tylko do odczytu.

zdarzenie komendy

Określa, czy zdarzenia komendy są generowane. Ten atrybut jest tylko do odczytu.

nazwa kolejki wejściowej komendy

Nazwa kolejki wejściowej komend systemowych. Ten atrybut jest tylko do odczytu.

poziom komendy

Poziom komendy obsługiwany przez menedżer kolejek. Ten atrybut jest tylko do odczytu.

Kontrola serwera komend

Określa, czy serwer komend powinien być uruchamiany automatycznie podczas uruchamiania menedżera kolejek. Ten atrybut jest tylko do odczytu.

Opcje połączenia

Opcje, które mają zastosowanie do metody **connect**. Wartością początkową jest MQCNO_NONE. W zależności od platformy mogą być możliwe następujące wartości dodatkowe:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_HANDLE_SHARE_NONE
- MQCNO_HANDLE_SHARE_BLOCK
- MQCNO_HANDLE_SHARE_NO_BLOCK
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_QSG

Identyfikator połączenia

Unikalny identyfikator, który umożliwia MQ niezawodne identyfikowanie aplikacji.

status połączenia

Wartość TRUE, jeśli jest połączona z menedżerem kolejek. Ten atrybut jest tylko do odczytu.

Znacznik połączenia

Znacznik, który ma zostać powiązany z połączeniem. Ten atrybut można ustawić tylko wtedy, gdy nie jest połączony. Początkowa wartość jest równa null.

Sprzęt szyfrujący

Szczegóły konfiguracji sprzętu szyfrującego. Dla połączeń klienta MQI produktu MQ .

nazwa kolejki niedostarczonych komunikatów

Nazwa kolejki niedostarczonych komunikatów. Ten atrybut jest tylko do odczytu.

domyślna nazwa kolejki transmisji

Domyślna nazwa kolejki transmisji. Ten atrybut jest tylko do odczytu.

listy dystrybucyjne

Możliwość obsługi menedżera kolejek w celu obsługi list dystrybucyjnych.

grupa dns

Nazwa grupy, do której należy dołączyć program nasłuchujący TCP obsługujący transmisje przychodzące dla grupy współużytkowania kolejek przy użyciu obsługi usług dynamicznych nazw domen programu Workload Manager. Ten atrybut jest tylko do odczytu.

dns wlm

Określa, czy program nasłuchujący TCP obsługujący transmisje przychodzące dla grupy współużytkowania kolejki powinien zarejestrować się w programie Workload Manager for Dynamic Domain Name Services. Ten atrybut jest tylko do odczytu.

pierwszy rekord uwierzytelniania

Pierwszy z jednego lub większej liczby obiektów klasy ImqAuthenticationRecord, w żadnym konkretnym porządku, w którym odwołanie do połączenia ImqAuthentication(ImqAuthentication) odnosi się do tego obiektu. Dla połączeń klienta MQI produktu MQ .

pierwszy obiekt zarządzany

Pierwszy z jednego lub większej liczby obiektów klasy ImqObject, w żadnym konkretnym porządku, w którym ImqObject **odwołanie do połączenia** odnosi się do tego obiektu. Wartością początkową jest zero.

Hamuj zdarzenie

Elementy sterujące hamują zdarzenia. Ten atrybut jest tylko do odczytu.

Wersja adresu IP

Który protokół IP (IPv4 lub IPv6) ma być używany dla połączenia kanału. Ten atrybut jest tylko do odczytu.

repozytorium kluczy

Położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty. Dla połączeń IBM MQ MQI client .

licznik resetowania klucza

Liczba niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed renegocjacją klucza tajnego. Ten atrybut ma zastosowanie tylko do połączeń klientów korzystających z produktu MQCONN. Patrz także liczba resetowanych kluczy ssl.

Zegar nasłuchiwanie

Odstęp czasu (w sekundach) między kolejnymi próbami zrestartowania obiektu nasłuchiwanie przez program IBM MQ , jeśli wystąpiła awaria APPC lub TCP/IP. Ten atrybut jest tylko do odczytu.

zdarzenie lokalne

Steruje zdarzeniami lokalnymi. Ten atrybut jest tylko do odczytu.

zdarzenie programu rejestrującego

Określa, czy generowane są zdarzenia dziennika odtwarzania. Ten atrybut jest tylko do odczytu.

Nazwa grupy LU

Ogólna nazwa LU, która powinna być używana przez proces nasłuchujący LU 6.2 obsługujący transmisje przychodzące dla grupy współużytkowania kolejki. Ten atrybut jest tylko do odczytu.

Nazwa LU

Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 . Ten atrybut jest tylko do odczytu.

Przyrostek ramienia lu62

Przyrostek SYS1.PARMLIB składowa APPCPMxx, która mianuje LUADD dla tego inicjatora kanału. Ten atrybut jest tylko do odczytu.

Kanały lu62

Maksymalna liczba kanałów, które mogą być bieżące lub klienci, które mogą być podłączone, które korzystają z protokołu transmisji LU 6.2 . Ten atrybut jest tylko do odczytu.

maksymalna liczba aktywnych kanałów

Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie. Ten atrybut jest tylko do odczytu.

Maksimum kanałów

Maksymalną liczbę kanałów bieżących (w tym kanałów połączenia z serwerem z połączonymi klientami). Ten atrybut jest tylko do odczytu.

maksymalne uchwyt

Maksymalna liczba uchwytów. Ten atrybut jest tylko do odczytu.

Maksymalna długość komunikatu

Maksymalna możliwa długość dla dowolnego komunikatu w dowolnej kolejce zarządzanej przez tego menedżera kolejek. Ten atrybut jest tylko do odczytu.

maksymalny priorytet

Maksymalny priorytet komunikatu. Ten atrybut jest tylko do odczytu.

Maks. liczba niezatw. kom.

Maksymalna liczba niezatwierdzonych komunikatów w jednostce lub pracy. Ten atrybut jest tylko do odczytu.

Rozliczanie MQI

Steruje kolekcjonowaniem informacji rozliczeniowych dla danych MQI. Ten atrybut jest tylko do odczytu.

Statystyka MQI

Steruje kolekcjonowaniem informacji monitorowania statystyk dla menedżera kolejek. Ten atrybut jest tylko do odczytu.

maksymalny port wychodzący

Wyższy koniec zakresu numerów portów, które mają być używane podczas wiązania kanałów wychodzących. Ten atrybut jest tylko do odczytu.

minimum portu wychodzącego

Dolny koniec zakresu numerów portów, które mają być używane podczas wiązania kanałów wychodzących. Ten atrybut jest tylko do odczytu.

Hasło

hasło powiązane z identyfikatorem użytkownika

zdarzenie dotyczące wydajności

Kontroluje zdarzenia wydajności. Ten atrybut jest tylko do odczytu.

platforma

Platforma, na której znajduje się menedżer kolejek. Ten atrybut jest tylko do odczytu.

Rozliczanie kolejek

Steruje kolekcjonowaniem informacji rozliczeniowych dla kolejek. Ten atrybut jest tylko do odczytu.

Monitorowanie kolejek

Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek. Ten atrybut jest tylko do odczytu.

Statystyka kolejek

Steruje kolekcjonowaniem danych statystycznych dla kolejek. Ten atrybut jest tylko do odczytu.

Limit czasu odbierania

W przybliżeniu, jak długo kanał komunikatów TCP/IP będzie oczekiwać na otrzymywanie danych, w tym pulsy, od partnera, przed powrotem do stanu nieaktywnego. Ten atrybut jest tylko do odczytu.

minimum limitu czasu odbierania

Minimalny czas oczekiwania przez kanał TCP/IP na odebranie danych, w tym pulsy, od partnera, przed powrotem do stanu nieaktywnego. Ten atrybut jest tylko do odczytu.

Typ limitu czasu odbierania

Kwalifikator stosowany do **limitu czasu odbierania**. Ten atrybut jest tylko do odczytu.

zdarzenie zdalne

Steruje zdarzeniami zdalnymi. Ten atrybut jest tylko do odczytu.

Nazwa repozytorium

Nazwa repozytorium. Ten atrybut jest tylko do odczytu.

Lista nazw repozytorium

Nazwa listy nazw repozytorium. Ten atrybut jest tylko do odczytu.

nazwa menedżera kolejek współużytkowanych

Określa, czy komenda MQOPENs kolejki współużytkowanej, w której nazwa ObjectQMgr jest nazwą innego menedżera kolejek w grupie współużytkowania kolejki, powinna zostać rozstrzygnięta na otwartą kolejkę współużytkowaną w lokalnym menedżerze kolejek. Ten atrybut jest tylko do odczytu.

Zdarzenie ssl

Określa, czy zdarzenia SSL są generowane. Ten atrybut jest tylko do odczytu.

Wymagane SSL FIPS

Określa, czy tylko algorytmy certyfikowane przez FIPS powinny być używane, jeśli kryptografia jest wykonywana w oprogramowaniu IBM MQ. Ten atrybut jest tylko do odczytu.

Licznik zerowania klucza SSL

Liczba niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed renowacją klucza tajnego. Ten atrybut jest tylko do odczytu.

początkowe zdarzenie zatrzymania

Steruje zdarzeniami uruchamiania. Ten atrybut jest tylko do odczytu.

Interwał statystyki

Jak często dane statystyczne są zapisywane w kolejce monitorowania. Ten atrybut jest tylko do odczytu.

Dostępność punktu synchronizacji

Dostępność udziału w punkcie synchronizacji. Ten atrybut jest tylko do odczytu.

Uwaga: Globalne jednostki pracy koordynowane przez menedżera kolejek nie są obsługiwane na platformie IBM i.  Można programować jednostkę pracy, koordynowaną zewnętrznym przez produkt IBM i, używając rodzimych wywołań systemowych _Rcommit i _Rback. Uruchomienie tego typu jednostki pracy przez uruchomienie aplikacji IBM MQ w ramach kontroli transakcji na poziomie zadania za pomocą komendy STRCMTCTL. Więcej szczegółowych informacji na ten temat zawiera sekcja Interfejsy do zewnętrznego menedżera punktów synchronizacji produktu IBM i. Opcje **Backout** i **commit** są obsługiwane na platformie IBM i dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek.

kanaly tcp

Maksymalna liczba kanałów, które mogą być bieżące lub klienci, które mogą być podłączone, które korzystają z protokołu transmisji TCP/IP. Ten atrybut jest tylko do odczytu.

Sprawdzanie połączenia TCP

Określa, czy narzędzie TCP KEEPALIVE ma być używane do sprawdzania, czy drugi koniec połączenia jest nadal dostępny. Ten atrybut jest tylko do odczytu.

Nazwa TCP

Nazwa jedyne lub domyślnego systemu TCP/IP, który ma być używany, w zależności od wartości **tcp stack type** (Typ stosu tcp). Ten atrybut jest tylko do odczytu.

Typ stosu TCP

Określa, czy inicjator kanału może używać tylko przestrzeni adresowej TCP/IP określonej w parametrze **tcp name**, czy może wiązać się z dowolnym wybranym adresem TCP/IP. Ten atrybut jest tylko do odczytu.

Zapis śledzenia trasy

Steruje rejestrowaniem informacji o śledzeniu trasy. Ten atrybut jest tylko do odczytu.

Interwał wyzwalacza

Przedział czasu wyzwalacza. Ten atrybut jest tylko do odczytu.

ID użytkownika

Na platformach UNIX i Linux rzeczywisty identyfikator użytkownika aplikacji. Na platformach Windows identyfikator użytkownika aplikacji.

Konstruktory

ImqQueueManager ();

Konstruktor domyślny.

Menedżer ImqQueue(const ImqQueueManager & *menedżer*);

Konstruktor kopiowania. **Status połączenia** będzie mieć wartość FALSE.

ImqQueueManager (const char * *nazwa*);

Ustawia parametr ImqObject **nazwa** na *nazwa*.

Destruktory

Jeśli obiekt ImqQueueManager zostanie zniszczony, zostanie automatycznie rozłączony.

Metody klasy (publiczne)

statyczne zachowanie MQLONG ();

Zwraca **zachowanie**.

void setBehavior(const MQLONG *zachowanie* = 0);

Ustawia **zachowanie**.

Metody obiektów (publiczne)

void operator = (const ImqQueueManager & *mgr*);

W razie potrzeby odłącza się i kopiuje dane instancji z *mgr*. **Status połączenia** ma wartość FALSE.

ImqBoolean accountingConnOverride (MQLONG & *statint*);

Udostępnia kopię wartości nadpisania połączeń rozliczeniowych. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG accountingConnOverride ();

Zwraca wartość nadpisania połączeń rozliczeniowych bez wskazania ewentualnych błędów.

ImqBoolean accountingInterval (MQLONG & *statint*);

Udostępnia kopię wartości przedziału czasu rozliczania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG accountingInterval ();

Zwraca wartość przedziału czasu rozliczania bez wskazania ewentualnych błędów.

ImqBoolean activityRecording (MQLONG & *rec*);

Udostępnia kopię wartości zapisu działania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG activityRecording ();

Zwraca wartość zapisu działania bez wskazania możliwych błędów.

ImqBoolean adoptNewMCACheck (MQLONG & *check*);

Udostępnia kopię wartości sprawdzania adopcji nowego MCA. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG adoptNewMCACheck ();

Zwraca wartość sprawdzania adopcji nowego MCA bez wskazania ewentualnych błędów.

ImqBoolean adoptNewMCAType (MQLONG & *type*);

Udostępnia kopię typu adopcji nowego agenta MCA. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG adoptNewMCAType ();

Zwraca typ nowego agenta MCA bez wskazania ewentualnych błędów.

QLONG authenticationType () const;

Zwraca typ uwierzytelniania.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);

Ustawia typ uwierzytelniania.

ImqBoolean authorityEvent(MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia uprawnień**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG authorityEvent();

Zwraca stan włączenia **zdarzenia uprawnień** bez wskazania ewentualnych błędów.

ImqBoolean backout ();

Wycofuje niezatwierdzone zmiany. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean -początek ();

Rozpoczyna jednostkę pracy. **Opcje rozpoczęcia** wpływają na zachowanie tej metody.

Zwraca wartość PRAWDA, jeśli powiedzie się, ale zwraca także wartość PRAWDA, nawet jeśli bazowa wywołanie MQBEGIN zwraca wartość MQRC_NO_EXTERNAL_UCZESTNIKÓW lub MQRC_W_IPANT_NOT_AVAILABLE (oba te elementy są powiązane z wartością MQCC_WARNING).

MQLONG beginOptions() const;

Zwraca **opcje rozpoczęcia**.

void setBeginOptions (const MQLONG opcje = MQBO_NONE);

Ustawia **opcje rozpoczęcia**.

ImqBoolean bridgeEvent (MQLONG & event);

Udostępnia kopię wartości zdarzenia mostu. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG bridgeEvent ();

Zwraca wartość zdarzenia mostu bez wskazania ewentualnych błędów.

ImqBoolean channelAutoDefinicja (MQLONG & wartość);

Udostępnia kopię wartości **automatycznej definicji kanału** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelAuto-definicja ();

Zwraca wartość **auto definicji kanału** bez wskazania ewentualnych błędów.

ImqBoolean channelAutoDefinitionEvent(MQLONG & wartość);

Udostępnia kopię wartości **zdarzenia automatycznego definiowania kanału** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelAutoDefinitionEvent();

Zwraca wartość **zdarzenie automatycznego definiowania kanału** bez wskazania możliwych błędów.

ImqBoolean channelAutoDefinitionExit(ImqString & nazwa);

Udostępnia kopię nazwy **wyjścia automatycznego definiowania kanału** . Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString channelAutoDefinitionExit();

Zwraca nazwę **wyjścia automatycznego definiowania kanału** bez wskazania ewentualnych błędów.

ImqBoolean channelEvent (MQLONG & event);

Udostępnia kopię wartości zdarzenia kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelEvent();

Zwraca wartość zdarzenia kanału bez wskazania ewentualnych błędów.

Adaptory MQLONG channelInitiator();

Zwraca wartość adapterów inicjatora kanału bez wskazania ewentualnych błędów.

ImqBoolean Adaptory channelInitiator(MQLONG & adapters);

Udostępnia kopię wartości adapterów inicjatora kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

Wywołanie funkcji MQLONG channelInitiator();

Zwraca wartość uruchomieniową inicjatora kanału bez wskazania ewentualnych błędów.

ImqBoolean channelInitiatorControl (MQLONG & init);

Udostępnia kopię wartości uruchamiania elementu sterującego inicjatora kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

Program rozsyłający MQLONG channelInitiator();

Zwraca wartość przekaźników inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorProgram rozsyłający (MQLONG & dispatchers);

Udostępnia kopię wartości programów rozsyłających inicjatora kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelInitiatorTraceAutoStart ();

Zwraca wartość automatycznego startu śledzenia inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);

Udostępnia kopię wartości automatycznego uruchamiania śledzenia inicjatora kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelInitiatorTraceTableSize ();

Zwraca wartość wielkości tabeli śledzenia inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorTraceTableSize (MQLONG & size);

Udostępnia kopię wartości wielkości tabeli śledzenia inicjatora kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean channelMonitoring (MQLONG & monchl);

Udostępnia kopię wartości monitorowania kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelMonitoring ();

Zwraca wartość monitorowania kanału bez wskazania ewentualnych błędów.

ImqBoolean channelReference(ImqChannel * & pchannel);

Udostępnia kopię **odwołania do kanału**. Jeśli **odwołanie do kanału** jest niepoprawne, ustawia parametr *pchannel* na wartość NULL. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqChannel * channelReference();

Zwraca **odwołanie do kanału** bez wskazania ewentualnych błędów.

ImqBoolean setChannelReference (ImqChannel & kanał);

Ustawia **odwołanie kanału**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setChannelReference (ImqChannel * kanał = 0);

Ustawia lub resetuje **odwołanie do kanału**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean channelStatistics (MQLONG & statchl);

Udostępnia kopię wartości statystyki kanału. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG channelStatistics ();

Zwraca wartość statystyki kanału bez wskazania ewentualnych błędów.

ImqBoolean characterSet(MQLONG & ccscid);

Udostępnia kopię **zestawu znaków**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG characterSet();

Zwraca kopię **zestawu znaków** bez wskazania ewentualnych błędów.

Liczba operacji MQLONG clientSslKeyReset() const;

Zwraca wartość licznika resetowania klucza SSL używaną w połączeniach klienta.

void setClientSslKeyResetCount(const MQLONG count);

Ustawia licznik resetowania klucza SSL używany w połączeniach klienta.

ImqBoolean clusterSenderMonitoring (MQLONG & monacsl);

Udostępnia kopię domyślnej wartości monitorowania nadajnika klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterSenderMonitorowanie ();

Zwraca wartość domyślną monitorowania nadajnika klastra bez wskazania ewentualnych błędów.

ImqBoolean clusterSenderStatystyka (MQLONG & statacsl);

Udostępnia kopię wartości statystyki nadawcy klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterSender-statystyki ();

Zwraca wartość statystyki nadawcy klastra bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkloadData (ImqString & data);

Udostępnia kopię **danych wyjścia obciążenia klastra**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString clusterWorkloadData ();

Zwraca **dane wyjścia obciążenia klastra** bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkloadWyjście (ImqString & nazwa);

Udostępnia kopię **nazwy wyjścia obciążenia klastra**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString clusterWorkloadExit ();

Zwraca **nazwę wyjścia obciążenia klastra** bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkloadLength (MQLONG & długość);

Udostępnia kopię **długości obciążenia klastra**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkloadLength ();

Zwraca **długość obciążenia klastra** bez wskazania możliwych błędów.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);

Udostępnia kopię obciążenia klastra, które ostatnio używa wartości kanałów. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkLoadMRU ();

Zwraca obciążenie klastra ostatnio używane przez wartość kanałów bez wskazania ewentualnych błędów.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);

Udostępnia kopię wartości kolejki użycia obciążenia klastra. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG clusterWorkLoadUseQ ();

Zwraca wartość kolejki wykorzystania obciążenia klastra bez wskazania ewentualnych błędów.

ImqBoolean commandEvent (MQLONG & event);

Udostępnia kopię wartości zdarzenia komendy. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG commandEvent ();

Zwraca wartość zdarzenia komendy bez wskazania ewentualnych błędów.

ImqBoolean commandInputQueueName(ImqString & nazwa);

Udostępnia kopię **nazwy kolejki wejściowej komend**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString commandInputQueueName();

Zwraca **nazwę kolejki wejściowej komendy** bez wskazania ewentualnych błędów.

ImqBoolean commandLevel(MQLONG & poziom);

Udostępnia kopię **poziomu komendy**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG commandLevel();

Zwraca **poziom komendy** bez wskazania ewentualnych błędów.

MQLONG commandServerControl ();

Zwraca wartość uruchamiania serwera komend bez wskazania ewentualnych błędów.

ImqBoolean commandServerControl (MQLONG & server);

Udostępnia kopię wartości uruchamiania elementu sterującego serwera komend. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean commit ();

Zatwierdza niezatwierdzone zmiany. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean connect ();

Łączy się z menedżerem kolejek z podanym ImqObject **nazwa**, domyślnym menedżerem kolejek jest lokalny menedżer kolejek. Aby nawiązać połączenie z konkretnym menedżerem kolejek, przed połączeniem należy użyć metody ImqObject **setName** . Jeśli istnieje **odwołanie do kanału**, jest on używany do przekazywania informacji o definicji kanału do tabeli MQCONN na dysku MQCD. Typ ChannelType w tabeli MQCD jest ustawiony na wartość MQCHT_CLNTCONN. Informacje o **odwołaniu do kanału** , które mają znaczenie tylko dla połączeń klientów, są ignorowane w przypadku połączeń

z serwerem. **Opcje łączenia** wpływają na zachowanie tej metody. Ta metoda ustawia **status połączenia** na wartość TRUE, jeśli jest ona pomyślna. Zwraca nowy status połączenia.

Jeśli istnieje pierwszy rekord uwierzytelniania, łańcuch rekordów uwierzytelniania jest używany do uwierzytelniania certyfikatów cyfrowych dla bezpiecznych kanałów klienta.

Można połączyć więcej niż jeden obiekt menedżera kolejek ImqQueuez tym samym menedżerem kolejek. Należy użyć tego samego uchwytu połączenia MQHCONN i udostępnić do współużytkowania funkcję jednostki pracy dla połączenia powiązanego z wątkiem. Pierwszy menedżer ImqQueuew celu nawiązania połączenia uzyskuje uchwyt MQHCONN. Ostatni menedżer ImqQueue, który ma zostać odłączony, wykonuje operację MQDISC.

W przypadku programu wielowątkowego zaleca się, aby dla każdego wątku był używany oddzielny obiekt ImqQueueManager.

ImqBinary connectionId () const;

Zwraca identyfikator połączenia.

ImqBinary connectionTag () const;

Zwraca **znacznik połączenia**.

ImqBoolean setConnectionTag (const MQBYTE128 *znacznik* = 0);

Ustawia **znacznik połączenia**. Jeśli parametr *znacznik* ma wartość zero, wyczyści **znacznik połączenia**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setConnectionTag (const ImqBinary & *znacznik*);

Ustawia **znacznik połączenia**. **Długość danych** elementu *znacznik* musi być równa zero (aby usunąć zaznaczenie pola **znacznik połączenia**). lub MQ_CONN_TAG_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG connectOptions() const;

Zwraca **opcje połączenia**.

void setConnectOptions (const MQLONG *opcje* = MQCNO_NONE);

Ustawia **opcje połączenia**.

ImqBoolean connectionStatus() const;

Zwraca **status połączenia**.

ImqString cryptographicHardware ();

Zwraca **sprzęt szyfrujący**.

ImqBoolean setCryptographicHardware (const char * *sprzęt* = 0);

Ustawia **sprzęt szyfrujący**. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean deadLetterQueueName(ImqString & *nazwa*);

Udostępnia kopię **nazwy kolejki niedostarczonych komunikatów**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString deadLetterQueueName();

Zwraca kopię **nazwy kolejki niedostarczonych komunikatów**, bez wskazania ewentualnych błędów.

ImqBoolean defaultTransmissionQueueName(ImqString & *nazwa*);

Udostępnia kopię **domyślnej nazwy kolejki transmisji**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString defaultTransmissionQueueName();

Zwraca **domyślną nazwę kolejki transmisji** bez wskazania ewentualnych błędów.

ImqBoolean rozłączenie ();

Rozłącza się z menedżerem kolejek i ustawia **status połączenia** na wartość FALSE. Powoduje zamknięcie wszystkich obiektów ImqProcess i ImqQueue powiązanych z tym obiektem, a następnie należy usunąć ich **odwołanie do połączenia** przed rozłączonym połączeniem. Jeśli więcej niż jeden obiekt ImqQueueManager jest połączony z tym samym menedżerem kolejek, to tylko ostatnie rozłączenie wykonuje fizyczne rozłączenie; inne wykonują logiczne rozłączenie. Niezatwierdzone zmiany są zatwierdzane tylko w przypadku fizycznego rozłączenia.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się. Jeśli jest wywoływana, gdy nie ma istniejącego połączenia, kod powrotu jest również prawdziwy.

ImqBoolean distributionLists(MQLONG & *obsługa*);

Udostępnia kopię wartości **list dystrybucyjnych** . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG distributionLists();

Zwraca wartość **list dystrybucyjnych** bez wskazania ewentualnych błędów.

ImqBoolean dnsGroup (ImqString & group);

Udostępnia kopię nazwy grupy DNS. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString dnsGroup ();

Zwraca nazwę grupy DNS bez wskazania ewentualnych błędów.

ImqBoolean dnsWlm (MQLONG & wlm);

Udostępnia kopię wartości menedżera DNS WLM. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG dnsWlm ();

Zwraca wartość DNS WLM bez wskazania ewentualnych błędów.

Rekord ImqAuthenticationRecord * firstAuthenticationRecord () const;

Zwraca **pierwszy rekord uwierzytelniania**.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * *air* = 0);

Ustawia **pierwszy rekord uwierzytelniania**.

ImqObject * firstManagedObject () const;

Zwraca **pierwszy obiekt zarządzany**.

ImqBoolean inhibitEvent(MQLONG & *zdarzenie*);

Udostępnia kopię stanu włączenia zdarzenia **zablokuj zdarzenie**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG inhibitEvent();

Zwraca stan włączenia **zdarzenia zablokowanej pracy** bez wskazania ewentualnych błędów.

ImqBoolean ipAddressWersja (MQLONG & version);

Udostępnia kopię wartości wersji adresu IP. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG ipAddressWersja ();

Zwraca wartość wersji adresu IP bez wskazania ewentualnych błędów.

ImqBoolean keepAlive (MQLONG & keepalive);

Udostępnia kopię wartości podtrzymanej. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG keepAlive ();

Powoduje zwrócenie wartości podtrzymanej bez wskazania ewentualnych błędów.

ImqString keyRepository ();

Zwraca **repozytorium kluczy**.

ImqBoolean setKeyRepository (const char * *repozytorium* = 0);

Ustawia **repozytorium kluczy**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean listenerTimer (MQLONG & timer);

Udostępnia kopię wartości licznika czasu nasłuchiwania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG listenerTimer ();

Zwraca wartość licznika czasu nasłuchiwania bez wskazania możliwych błędów.

ImqBoolean localEvent(MQLONG & *zdarzenie*);

Udostępnia kopię stanu włączenia **zdarzenia lokalnego**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG localEvent();

Zwraca stan włączenia **zdarzenia lokalnego** bez wskazania ewentualnych błędów.

ImqBoolean loggerEvent (MQLONG & count);

Udostępnia kopię wartości zdarzenia programu rejestrującego. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG loggerEvent ();

Zwraca wartość zdarzenia programu rejestrującego bez wskazania ewentualnych błędów.

ImqBoolean luGroupNazwa (ImqString & name);

Udostępnia kopię nazwy grupy LU. Zwraca wartość PRAWDA, jeśli powiodła się

ImqString luGroupNazwa ();

Zwraca nazwę grupy LU bez wskazania ewentualnych błędów.

ImqBoolean lu62ARMSuffix (ImqString & suffix);

Udostępnia kopię przyrostka ARM LU62 . Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString lu62ARMSuffix ();

Zwraca przyrostek ARM LU62 bez wskazania ewentualnych błędów.

ImqBoolean luName (ImqString & name);

Udostępnia kopię nazwy jednostki logicznej. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString luName ();

Zwraca nazwę jednostki logicznej bez wskazania ewentualnych błędów.

ImqBoolean maximumActiveKanały (MQLONG i kanały);

Udostępnia kopię wartości maksymalnej liczby aktywnych kanałów. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumActiveKanały ();

Zwraca wartość maksymalnej liczby aktywnych kanałów bez wskazania ewentualnych błędów.

ImqBoolean maximumCurrentKanały (MQLONG i kanały);

Udostępnia kopię wartości maksymalnej liczby bieżących kanałów. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumCurrentKanały ();

Zwraca wartość maksymalnej liczby bieżących kanałów bez wskazania ewentualnych błędów.

ImqBoolean maximumHandles(MQLONG & liczba);

Udostępnia kopię **maksymalnej liczby uchwytów**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumHandles();

Zwraca **maksymalną liczbę uchwytów** bez wskazania możliwych błędów.

ImqBoolean maximumLu62Channels (MQLONG & kanały);

Udostępnia kopię maksymalnej wartości kanałów LU62 . Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumLu62Channels ();

Zwraca maksymalną wartość kanałów LU62 bez wskazania ewentualnych błędów.

ImqBoolean maximumMessageLength (MQLONG & długość);

Udostępnia kopię **maksymalnej długości komunikatu**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumMessageLength ();

Zwraca **maksymalną długość komunikatu** bez wskazania ewentualnych błędów.

ImqBoolean maximumPriority(MQLONG & priorytet);

Udostępnia kopię **maksymalnego priorytetu**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumPriority();

Zwraca kopię **maksymalnego priorytetu** bez wskazania ewentualnych błędów.

ImqBoolean maximumTcpKanały (MQLONG i kanały);

Udostępnia kopię maksymalnej wartości kanałów TCP. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumTcpKanały ();

Zwraca maksymalną wartość kanałów TCP bez wskazania ewentualnych błędów.

ImqBoolean maximumUncommittedKomunikaty (MQLONG & liczba);

Udostępnia kopię **maksymalnej liczby niezatwierdzonych komunikatów**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG maximumUncommittedKomunikaty ();

Zwraca **maksymalną liczbę niezatwierdzonych komunikatów** bez wskazania możliwych błędów.

ImqBoolean mqiAccounting (MQLONG & statint);

Udostępnia kopię wartości rozliczania MQI. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG mqiAccounting ();

Zwraca wartość rozliczania MQI bez wskazania ewentualnych błędów.

ImqBoolean mqStatistics (MQLONG & statmq);

Udostępnia kopię wartości statystyki MQI. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG mqStatistics ();

Zwraca wartość statystyki MQI bez wskazania ewentualnych błędów.

ImqBoolean outboundPortMax (MQLONG & max);

Udostępnia kopię maksymalnej wartości portu wychodzącego. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG outboundPortMaks. ();

Zwraca maksymalną wartość portu wychodzącego bez wskazania ewentualnych błędów.

ImqBoolean outboundPortMin. (MQLONG & min);

Udostępnia kopię minimalnej wartości portu wychodzącego. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG outboundPortMin ();

Zwraca minimalną wartość portu wychodzącego bez wskazania ewentualnych błędów.

Hasło ImqBinary () const;

Zwraca hasło używane w połączeniach klienta.

ImqBoolean setPassword (const ImqString & password);

Ustawia hasło używane w połączeniach klienckich.

ImqBoolean setPassword (const char * = 0 hasło);

Ustawia hasło używane w połączeniach klienckich.

ImqBoolean setPassword (const ImqBinary & password);

Ustawia hasło używane w połączeniach klienckich.

ImqBoolean performanceEvent(MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia wydajności**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG performanceEvent();

Zwraca stan włączenia **zdarzenia wydajności** bez wskazania ewentualnych błędów.

Platforma ImqBoolean (MQLONG & platforma);

Udostępnia kopię **platformy**. Zwraca wartość PRAWDA, jeśli powiodła się.

platforma MQLONG ();

Zwraca **platformę** bez wskazania ewentualnych błędów.

ImqBoolean queueAccounting (MQLONG & acctq);

Udostępnia kopię wartości rozliczania kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueAccounting ();

Zwraca wartość księgową kolejki bez wskazania ewentualnych błędów.

ImqBoolean queueMonitoring (MQLONG & monq);

Udostępnia kopię wartości monitorowania kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueMonitoring ();

Zwraca wartość monitorowania kolejki bez wskazania ewentualnych błędów.

ImqBoolean queueStatistics (MQLONG & statq);

Udostępnia kopię wartości statystyki kolejki. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG queueStatistics ();

Zwraca wartość statystyki kolejki bez wskazania ewentualnych błędów.

ImqBoolean receiveTimeout (MQLONG & timeout);

Udostępnia kopię wartości limitu czasu odbierania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG receiveTimeout ();

Zwraca wartość limitu czasu odbierania bez wskazania możliwych błędów.

ImqBoolean receiveTimeoutMin (MQLONG & min);

Udostępnia kopię minimalnej wartości limitu czasu odbierania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG receiveTimeoutMin ();

Zwraca minimalną wartość limitu czasu odbierania bez wskazania możliwych błędów.

ImqBoolean receiveTimeoutType (MQLONG & type);

Udostępnia kopię typu limitu czasu odbierania. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG receiveTimeoutType ();

Zwraca typ limitu czasu odbierania bez wskazania ewentualnych błędów.

ImqBoolean remoteEvent(MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia zdalnego**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG remoteEvent();

Zwraca stan włączenia **zdarzenia zdalnego** bez wskazania ewentualnych błędów.

ImqBoolean repositoryName(ImqString & nazwa);

Udostępnia kopię **nazwy repozytorium**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString repositoryName();

Zwraca **nazwę repozytorium** bez wskazania ewentualnych błędów.

ImqBoolean repositoryNameListName (ImqString & nazwa);

Udostępnia kopię **nazwy listy nazw repozytorium**. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString repositoryNameListName ();

Zwraca kopię **nazwy listy nazw repozytorium** bez wskazania ewentualnych błędów.

ImqBoolean sharedQueueQueueManagerNazwa (MQLONG & nazwa);

Udostępnia kopię wartości nazwy menedżera kolejek współużytkowanych. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG sharedQueueQueueManagerName ();

Zwraca wartość nazwy menedżera kolejek współużytkowanych bez wskazania możliwych błędów.

ImqBoolean sslEvent (MQLONG & event);

Udostępnia kopię wartości zdarzenia SSL. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG sslEvent ();

Zwraca wartość zdarzenia SSL bez wskazania ewentualnych błędów.

ImqBoolean sslFips (MQLONG & sslfips);

Udostępnia kopię wartości SSL FIPS. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG sslFips ();

Zwraca wartość SSL FIPS bez wskazania ewentualnych błędów.

ImqBoolean sslKeyResetCount (MQLONG & count);

Udostępnia kopię wartości licznika resetowania klucza SSL. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG sslKeyResetCount ();

Zwraca wartość licznika resetowania klucza SSL bez wskazania ewentualnych błędów.

ImqBoolean startStopEvent (MQLONG & zdarzenie);

Udostępnia kopię stanu włączenia **zdarzenia początkowego zatrzymania**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG startStopEvent ();

Zwraca stan włączenia **zdarzenia początkowego zatrzymania** bez wskazania ewentualnych błędów.

ImqBoolean statisticsInterval (MQLONG & statint);

Udostępnia kopię wartości przedziału czasu statystyk. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG statisticsInterval ();

Zwraca wartość przedziału czasu statystyki bez wskazania ewentualnych błędów.

ImqBoolean syncPointDostępność (MQLONG & sync);

Udostępnia kopię wartości **syncpoint availability** (Dostępność punktu synchronizacji). Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG syncPointDostępność ();

Zwraca kopię wartości **syncpoint availability** , bez wskazania ewentualnych błędów.

ImqBoolean tcpName (ImqString & name);

Udostępnia kopię nazwy systemu TCP. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqString tcpName ();

Zwraca nazwę systemu TCP bez wskazania ewentualnych błędów.

ImqBoolean tcpStackTyp (MQLONG & type);

Udostępnia kopię typu stosu TCP. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG tcpStackTyp ();

Zwraca typ stosu TCP bez wskazania ewentualnych błędów.

ImqBoolean traceRouteRejestrowanie (MQLONG & routerec);

Udostępnia kopię wartości zapisu trasy śledzenia. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG traceRouteZapis ();

Zwraca wartość zapisu trasy śledzenia bez wskazania ewentualnych błędów.

ImqBoolean triggerInterval(MQLONG & interwał);

Udostępnia kopię **przedziału czasu wyzwacza**. Zwraca wartość PRAWDA, jeśli powiodła się.

MQLONG triggerInterval();

Zwraca **przedział czasu wyzwacza** bez wskazania możliwych błędów.

ImqBinary userId () const;

Zwraca identyfikator użytkownika używany w połączeniach klientów.

ImqBoolean setUserId (const ImqString & id);

Ustawia ID użytkownika używany w połączeniach klienckich.

ImqBoolean setUserId (const char * = 0 id);

Ustawia ID użytkownika używany w połączeniach klienckich.

ImqBoolean setUserId (const ImqBinary & id);

Ustawia ID użytkownika używany w połączeniach klienckich.

Metody obiektów (chronione)**void setFirstManagedObject (const ImqObject * obiekt = 0);**

Ustawia **pierwszy obiekt zarządzany**.

Dane obiektu (chronione)**MQHCONN ohconn**

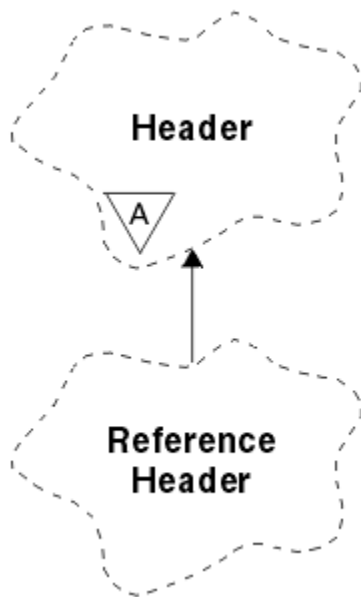
Uchwyt połączenia IBM MQ (znaczące tylko wtedy, gdy **status połączenia** ma wartość TRUE).

Kody przyczyny

- MQRC_ATTRIBUTE_LOCKED
- Błąd MQRC_ENVIRONMENT_ERROR
- MQRC_FUNCTION_NOT_SUPPORTED
- MQRC_REFERENCE_ERROR
- (kody przyczyny dla MQBACK)
- (kody przyczyny dla komendy MQBEGIN)
- (kody przyczyny dla MQCMIT)
- (kody przyczyny dla MQCONN)
- (kody przyczyny dla MQDISC)
- (kody przyczyny dla MQCONN)

Klasa ImqReferencenagłówka C++

Ta klasa hermetyzuje cechy struktury danych MQRMH.



Rysunek 66. Klasa nagłówka *ImqReference*

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji [“Odwołanie do nagłówka ImqReference”](#) na stronie 1937.

- [“Atrybuty obiektu”](#) na stronie 2020
- [“Konstruktory”](#) na stronie 2021
- [“Przeciążone metody ImqItem”](#) na stronie 2021
- [“Metody obiektów \(publiczne\)”](#) na stronie 2021
- [“Dane obiektu \(chronione\)”](#) na stronie 2022
- [“Kody przyczyny”](#) na stronie 2022

Atrybuty obiektu

środowisko docelowe

Środowisko dla miejsca docelowego. Wartością początkową jest łańcuch o wartości NULL.

Nazwa miejsca docelowego

Nazwa miejsca docelowego danych. Wartością początkową jest łańcuch o wartości NULL.

Identyfikator instancji

Identyfikator instancji. Wartość binarna (MQBYTE24) o długości MQ_OBJECT_INSTANCE_ID_LENGTH. Wartością początkową jest MQOII_NONE.

długość logiczna

Logiczne lub zamierzone, długość danych komunikatu, które są zgodne z tym nagłówkiem. Wartością początkową jest zero.

przesunięcie logiczne

Przesunięcie logiczne dla danych komunikatu, które mają być interpretowane w kontekście danych jako całości, w ostatecznym miejscu docelowym. Wartością początkową jest zero.

przesunięcie logiczne 2

Rozszerzenie o wysokiej kolejności do **przesunięcia logicznego**. Wartością początkową jest zero.

Typ odniesienia

Typ odniesienia. Wartością początkową jest łańcuch o wartości NULL.

Środowisko źródłowe

Środowisko dla źródła. Wartością początkową jest łańcuch o wartości NULL.

NAZWA ŹRÓDŁA

Nazwa źródła danych. Wartością początkową jest łańcuch o wartości NULL.

Konstruktory

ImqReferenceNagłówek ();

Konstruktor domyślny.

Nagłówek ImqReference(const ImqReferenceHeader & header);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & komunikat);

Wstawia strukturę danych MQRMH do buforu komunikatów na początku, dalej przesuwa istniejące dane komunikatu i ustawia wartość *msg format* na wartość MQFMT_REF_MSG_HEADER.

Więcej szczegółów można znaleźć w opisie metody klasy ImqHeader w systemie [“Klasa języka C++ ImqHeader” na stronie 1965](#).

virtual ImqBoolean pasteIn (ImqMessage & komunikat);

Odczytuje strukturę danych MQRMH z buforu komunikatów.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_REF_MSG_HEADER.

Więcej szczegółów można znaleźć w opisie metody klasy ImqHeader w systemie [“Klasa języka C++ ImqHeader” na stronie 1965](#).

Metody obiektów (publiczne)

void operator = (const ImqReferenceHeader & header);

Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqString destinationEnvironment () const ;

Zwraca kopię **środowiska docelowego**.

void setDestinationEnvironment (const char * środowisko = 0);

Ustawia **środowisko docelowe**.

ImqString destinationName () const ;

Zwraca kopię **nazwy miejsca docelowego**.

void setDestinationName (const char * nazwa = 0);

Ustawia **nazwę miejsca docelowego**.

ImqBinary instanceId () const ;

Zwraca kopię **identyfikatora instancji**.

ImqBoolean setInstanceId (const ImqBinary & id);

Ustawia **identyfikator instancji**. Wartość **długość danych** elementu *element* musi mieć wartość 0 lub wartość MQ_OBJECT_INSTANCE_ID_LENGTH. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

void setInstanceId (const MQBYTE24 id = 0);

Ustawia **identyfikator instancji**. Wartość *id* może być równa zero, która jest taka sama, jak wartość parametru MQOII_NONE. Jeśli parametr *id* ma wartość niezerową, musi mieć adres MQ_OBJECT_INSTANCE_ID_LENGTH bajtów danych binarnych. W przypadku korzystania z predefiniowanych wartości, takich jak MQOII_NONE, może być konieczne dokonanie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQOII_NONE.

MQLONG logicalLength () const ;

Zwraca **długość logiczną**.

void setLogicalLength (const MQLONG długość);

Ustawia **długość logiczną**.

MQLONG logicalOffset () const ;

Zwraca przesunięcie logiczne.

void setLogicalOffset (const MQLONG przesunięcie);

Ustawia przesunięcie logiczne.

MQLONG logicalOffset2 () const ;

Zwraca przesunięcie logiczne 2.

void setLogicalOffset2 (const MQLONG przesunięcie);

Ustawia przesunięcie logiczne 2.

ImqString referenceType () const ;

Zwraca kopię typu referencyjnego.

void setReferenceType (const char * nazwa = 0);

Ustawia typ odwołania.

ImqString sourceEnvironment () const ;

Zwraca kopię środowiska źródłowego.

void setSourceEnvironment (const char * środowisko = 0);

Ustawia środowisko źródłowe.

ImqString sourceName () const ;

Zwraca kopię nazwy źródła.

void setSourceName (const char * nazwa = 0);

Ustawia nazwę źródła.

Dane obiektu (chronione)

MQRMH omqrmh

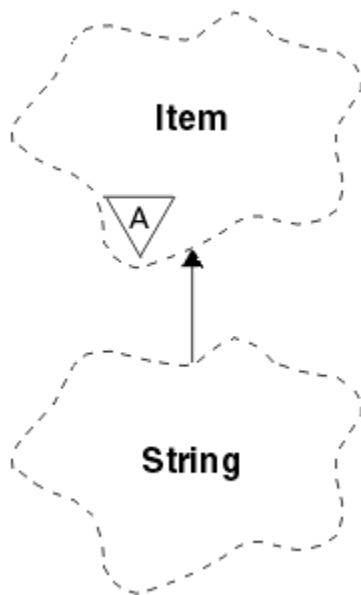
Struktura danych MQRMH.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- MQRC_STRUC_ID_BŁĄD
- MQRC_INSUFFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT
- Błąd MQRC_ENCODING_ERROR

Klasa ImqString C++

Ta klasa umożliwia przechowywanie łańcuchów znaków i manipulowanie w przypadku łańcuchów zakończonych znakiem o kodzie zero.



Rysunek 67. Klasa *ImqString*

Użyj parametru *ImqString* w miejsce znaku **char *** w większości sytuacji, w których parametr wywołuje znak **char ***.

- [“Atrybuty obiektu” na stronie 2023](#)
- [“Konstruktory” na stronie 2023](#)
- [“Metody klasy \(publiczne\)” na stronie 2024](#)
- [“Przeciążone metody *ImqItem*” na stronie 2024](#)
- [“Metody obiektów \(publiczne\)” na stronie 2024](#)
- [“Metody obiektów \(chronione\)” na stronie 2027](#)
- [“Kody przyczyny” na stronie 2027](#)

Atrybuty obiektu

znaków

Znaki w **pamięci masowej** poprzedzające końcowe wartości null.

długość

Liczba bajtów w **znakach**. Jeśli nie ma **pamięci**, **długość** wynosi zero. Wartością początkową jest zero.

pamięć masowa

Ulotna tablica bajtów o dowolnym rozmiarze. Końcowe wartości NULL muszą być zawsze obecne w **pamięci masowej** po **znakach**, tak aby można było wykryć koniec **znaków**. Metody dbają o to, aby ta sytuacja została zachowana, ale w przypadku bezpośredniego ustawienia bajtów w tablicy, że po modyfikacji istnieje końcowy znak o wartości NULL. Początkowo nie ma atrybutu **storage**.

Konstruktory

ImqString();

Konstruktor domyślny.

ImqString(const ImqString & łańcuch);

Konstruktor kopiowania.

ImqString(const char c);

Znaki zawierają c.

ImqString(const char * tekst);

Znaki są kopiowane z *tekstu*.

ImqString(const void * *buffer*, const size_t *długość*);

Kopiuje *długość* bajtów poczynając od *buforu* i przypisuje je do **znaków**. Podstawienie jest wykonywane dla wszystkich znaków o kodzie zero skopiowanych. Znakiem podstawienia jest kropka (.). Żadne inne znaki nie są brane pod uwagę przy kopiowaniu innych znaków niedrukowalnych lub niemożliwych do wyświetlenia.

Metody klasy (publiczne)

static ImqBoolean copy (char * *docelowy_bufor*, const size_t *długość*, const char * *źródła-bufor*, const char *pad* = 0);

Kopiuje do *długość* bajtów z *bufor-źródła* do *bufor-docelowy*. Jeśli liczba znaków w polu *bufor-źródłowy* jest niewystarczająca, wypełni pozostałe miejsce w polu *docelowy-bufor* znakami *dopełnienie*. *bufor-źródłowy* może być zerem. *docelowy-bufor* może być zerem, jeśli *długość* również wynosi zero. Wszelkie kody błędów są tracone. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

static ImqBoolean copy (char * *buffer-bufor*, const size_t *length*, const char * *source-buffer*, ImqError & *error-object*, const char *pad* = 0);

Kopiuje do *długość* bajtów z *bufor-źródła* do *bufor-docelowy*. Jeśli liczba znaków w polu *bufor-źródłowy* jest niewystarczająca, wypełni pozostałe miejsce w polu *docelowy-bufor* znakami *dopełnienie*. *bufor-źródłowy* może być zerem. *docelowy-bufor* może być zerem, jeśli *długość* również wynosi zero. Wszystkie kody błędów są ustawiane w katalogu *obiekt-błądu*. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);

Kopiuje **znaki** do buforu komunikatów, zastępując dowolną istniejącą treść. Ustawia wartość parametru *msg format* na wartość MQFMT_STRING.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*);

Ustawia **znaki**, przesyłając pozostałe dane z buforu komunikatów, zastępując istniejące **znaki**.

Aby możliwe było pomyślne, **kodowanie** obiektu *msg* musi mieć wartość MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT na wartość MQENC_NATIVE.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_STRING.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

Metody obiektów (publiczne)

char & operator [] (const size_t *przesunięcie*) const;

Odwołuje się do znaku w pozycji *przesunięcie* w **pamięci masowej**. Upewnij się, że odpowiedni bajt istnieje i jest adresowalny.

Operator ImqString () (const size_t *przesunięcie*, const size_t *długość* = 1) const;

Zwraca podłańcuch, kopiując bajty z **znaków** poczynając od pozycji *przesunięcie*. Jeśli *długość* wynosi zero, zwraca resztę **znaków**. Jeśli kombinacja *przesunięcie* i *długość* nie powoduje utworzenia odwołania w obrębie **znaków**, zwraca pusty łańcuch ImqString.

void operator = (const ImqString & *łańcuch*);

Kopiuje dane instancji z *łańcucha*, zastępując istniejące dane instancji.

Operator ImqString + (const char *c*) const;

Zwraca wynik dopisania *c* do **znaków**.

Operator ImqString + (znak const * tekst) const;

Zwraca wynik dopisania *tekst* do **znaków**. Może to być również odwrócone. Na przykład:

```
strOne + "string two" ;  
"string one" + strTwo ;
```

Uwaga: Chociaż większość kompilatorów akceptuje łańcuch **strOne + "string two"**; Microsoft Visual C++ wymaga **strOne + (char *) "string two"**;

Operator ImqString + (const ImqString & string1) const;

Zwraca wynik dopisania *string1* do **znaków**.

Operator ImqString + (const double liczba) const;

Zwraca wynik dopisania *liczba* do **znaków** po konwersji na tekst.

Operator ImqString + (const long liczba) const;

Zwraca wynik dopisania *liczba* do **znaków** po konwersji na tekst.

operator void + = (const char c);

Dopisuje znak *c* do **znaków**.

operator void + = (const char * tekst);

Dołącza *tekst* do **znaków**.

operator void + = (const ImqString & łańcuch);

Dołącza łańcuch *łańcuch* do **znaków**.

operator void + = (const double liczba);

Dopisuje *liczba* do **znaków** po konwersji na tekst.

void operator + = (const long liczba);

Dopisuje *liczba* do **znaków** po konwersji na tekst.

operator char * () const;

Zwraca adres pierwszego bajtu w **pamięci masowej**. Ta wartość może wynosić zero i jest ulotna. Tej metody należy używać tylko w celach tylko do odczytu.

Operator ImqBoolean < (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Wynik ma wartość PRAWDA, jeśli jest mniejsza niż i FAŁSZ, jeśli jest większa lub równa wartości.

Operator ImqBoolean > (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Wynik ma wartość PRAWDA, jeśli jest większy niż i FAŁSZ, jeśli jest mniejszy lub równy.

Operator ImqBoolean < = (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Wynik ma wartość TRUE, jeśli jest większy lub równy lub FALSE, jeśli jest większy niż.

Operator ImqBoolean > = (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Wynik ma wartość PRAWDA, jeśli jest większy lub równy lub FALSE, jeśli jest mniejszy niż.

Operator ImqBoolean == (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Zwraca wartość PRAWDA lub FAŁSZ.

Operator ImqBoolean != (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch* przy użyciu metody **porównaj**. Zwraca wartość PRAWDA lub FAŁSZ.

short compare (const ImqString & łańcuch) const;

Porównuje **znaki** z tymi z *łańcuch*. Wynik jest równy zero, jeśli **znaki** są równe, ujemne, jeśli są mniejsze niż i dodatnie, jeśli są większe niż. W porównaniu rozróżniana jest wielkość liter. Wartość ImqString o wartości NULL jest traktowana jako wartość mniejsza niż wartość ImqStringo wartości innej niż NULL.

ImqBoolean copyOut(char * buffer, const size_t length, const char pad = 0);

Kopiuje do *długość* bajtów z **znaków** do *buforu*. Jeśli liczba znaków **znaków** jest niewystarczająca, wypełni pozostałe miejsce w polu *bufor* znakami *dopełnienie*. Wartość *bufor* może wynosić zero, jeśli *długość* również wynosi zero. Zwraca wartość PRAWDA, jeśli powiodła się.

size_t copyOut(long & liczba) const;

Ustawia wartość *number* na **characters** po konwersji z tekstu i zwraca liczbę znaków biorących udział w konwersji. Jeśli wartość jest równa zero, konwersja nie została wykonana, a wartość *liczba* nie jest ustawiona. Sekwencja znaków przekształcalnych musi zaczynać się od następujących wartości:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & token, const char c = '') const;

Jeśli **znaki** zawierają jeden lub więcej znaków innych niż *c*, oznacza to, że jest to pierwsza ciągła sekwencja takich znaków. W tym przypadku element *token* jest ustawiony na tę sekwencję, a zwracana wartość jest sumą liczby wiodących znaków *c* i liczbą bajtów w sekwencji. W przeciwnym razie zwraca zero i nie ustawia wartości *token*.

size_t cutOut(long & liczba);

Ustawia wartość *number* jako metodę **copy**, ale usuwa również z **znaków** liczbę bajtów wskazanych przez wartość zwracaną. Na przykład łańcuch przedstawiony w poniższym przykładzie może zostać wycięty na trzy liczby za pomocą komendy **cutOut** (*liczba*). trzy razy:

```
strNumbers = "-1 0 +55 "
while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & token, const char c = '')

Ustawia *leksem* jako metodę **copyOut** i usuwa z **znaki** znaki *strToken*, a także wszystkie znaki *c*, które poprzedzają znaki *leksem*. Jeśli wartość *c* nie jest pusta, zostaną usunięte znaki *c*, które bezpośrednio powiodą się jako znaki *token*. Zwraca liczbę usuniętych znaków. Na przykład łańcuch przedstawiony w poniższym przykładzie może zostać wycięty na trzy elementy za pomocą komendy **cutOut** (*token*). trzy razy:

```
strText = " Program Version 1.1 "
while ( strText.cutOut( token ) );
// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

W poniższym przykładzie przedstawiono sposób analizowania nazwy ścieżki DOS:

```
strPath = "C:\OS2\BITMAP\OS2LOGO.BMP"
strPath.cutOut( strDrive, ':' );
strPath.stripLeading( ':' );
while ( strPath.cutOut( strFile, '\\' ) );
// strDrive becomes "C".
// strFile becomes "OS2", then "BITMAP",
// then "OS2LOGO.BMP" leaving strPath empty.
```

ImqBoolean find(const ImqString & łańcuch);

Wyszukuje dokładną zgodność dla *łańcucha* w dowolnym miejscu w obrębie **znaków**. Jeśli nie zostanie znalezione żadne dopasowanie, zwracana jest wartość FALSE. W przeciwnym razie zwraca wartość PRAWDA. Jeśli parametr *łańcuch* ma wartość NULL, zwraca wartość PRAWDA.

ImqBoolean find(const ImqString & string, size_t & offset);

Wyszukuje dokładną zgodność *łańcuch* gdzieś w obrębie **znaków** z przesunięcia *przesunięcie*. Jeśli parametr *łańcuch* ma wartość NULL, zwraca wartość PRAWDA bez aktualizowania wartości *przesunięcie*. Jeśli dopasowanie nie zostanie znalezione, zwracana jest wartość FALSE (wartość

przesunięcie mogła zostać zwiększona). Jeśli zostanie znaleziony zgodny element, zwróci wartość PRAWDA i zaktualizuje wartość *przesunięcie* na przesunięcie *łańcuch* w obrębie **znaków**.

rozmiar_t () const;

Zwraca **długość**.

ImqBoolean pasteIn(const double *liczba*, const char * *format* = "%f");

Dopisuje *liczba* do **znaków** po konwersji na tekst. Zwraca wartość PRAWDA, jeśli powiodła się.

Specyfikacja *format* jest używana do formatowania konwersji zmiennopozycyjnej. Jeśli zostanie podany, musi być on odpowiedni do użycia z **printf** i liczbą zmiennopozycyjną, na przykład **%3f**.

ImqBoolean pasteIn(const long *liczba*);

Dopisuje *liczba* do **znaków** po konwersji na tekst. Zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean pasteIn(const void * *buffer*, const size_t *długość*);

Dołącza *długość* bajtów z *buforu* do **znaków** i dodaje końcowe wartości null. Zastępuje wszystkie znaki null skopiowane. Znakiem podstawienia jest kropka (.). Żadne inne znaki nie są brane pod uwagę przy kopiowaniu innych znaków niedrukowalnych lub niewyświetlanych. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

Zestaw ImqBoolean (const char * *buffer*, const size_t *długość*);

Ustawia **znaki** z pola znakowego o stałej długości, które może zawierać wartość null. W razie potrzeby dopisuje wartość NULL do znaków z pola o stałej długości. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqBoolean setStorage(const size_t *długość*);

Przydziela (lub przydziela) **pamięć masową**. Zachowuje wszelkie oryginalne **znaki**, w tym wszystkie końcowe wartości null, jeśli nadal istnieje dla nich miejsce, ale nie inicjuje żadnej dodatkowej pamięci masowej.

Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

rozmiar_t pamięci () const;

Zwraca liczbę bajtów w **pamięci masowej**.

size_t stripLeading(const char *c* = " ");

Usuwa znaki wiodące *c* z **znaków** i zwraca liczbę usuniętych znaków.

size_t stripTrailing(const char *c* = " ");

Usuwa końcowe znaki *c* z **znaków** i zwraca liczbę usuniętych.

ImqString upperCase() const;

Zwraca wielką kopię **znaków**.

Metody obiektów (chronione)

ImqBoolean assign (const ImqString & *łańcuch*);

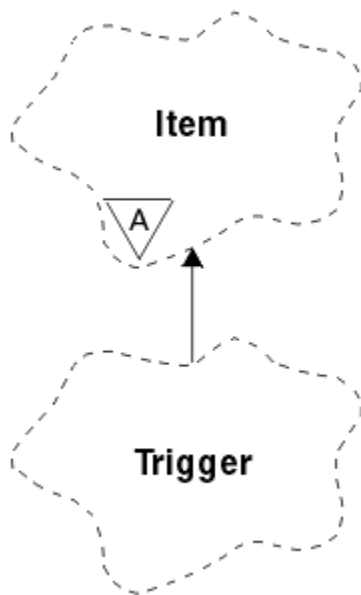
Odpowiada równoważnej metodzie **operator =**, ale nie jest to metoda wirtualna. Zwraca wartość PRAWDA, jeśli powiodła się.

Kody przyczyny

- MQRC_DATA_OBCIĘTY
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR-BŁĄD
- MQRC_INCONSISTENT_FORMAT

Klasa ImqTrigger C++

Ta klasa hermetykuje strukturę danych MQTM (komunikat wyzwalacza).



Rysunek 68. Klasa *ImqTrigger*

Obiekty tej klasy są zwykle używane przez program monitora wyzwalacza. Zadaniem programu monitorującego wyzwalacza jest oczekiwanie na te konkretne komunikaty i działanie na nich, aby zapewnić, że inne aplikacje produktu IBM MQ są uruchamiane, gdy komunikaty będą na nie czekać.

Przykład użycia można znaleźć w przykładowym programie IMQSTRG.

- [“Atrybuty obiektu” na stronie 2028](#)
- [“Konstruktory” na stronie 2029](#)
- [“Przeciążone metody ImqItem” na stronie 2029](#)
- [“Metody obiektów \(publiczne\)” na stronie 2029](#)
- [“Dane obiektu \(chronione\)” na stronie 2030](#)
- [“Kody przyczyny” na stronie 2030](#)

Atrybuty obiektu

Identyfikator aplikacji

Tożsamość aplikacji, która wysłała komunikat. Wartością początkową jest łańcuch o wartości NULL.

Typ aplikacji

Typ aplikacji, która wysłała komunikat. Wartością początkową jest zero. Możliwe są następujące wartości dodatkowe:

- MQAT_AIX
- MQAT_CICS
- MQAT_DOS
- MQAT_IMS
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS

- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

Dane środowiska

Dane środowiska dla procesu. Wartością początkową jest łańcuch o wartości NULL.

Nazwa procesu

Nazwa procesu. Wartością początkową jest łańcuch o wartości NULL.

Nazwa kolejki

Nazwa kolejki, która ma zostać uruchomiona. Wartością początkową jest łańcuch o wartości NULL.

Dane wyzwalacza

Wyzwalanie danych dla procesu. Wartością początkową jest łańcuch o wartości NULL.

Dane użytkownika

Dane użytkownika dla procesu. Wartością początkową jest łańcuch o wartości NULL.

Konstruktory

ImqTrigger();

Konstruktor domyślny.

ImqTrigger(const ImqTrigger & wyzwalacz);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & komunikat);

Zapisuje strukturę danych MQTM w buforze komunikatów, zastępując dowolną istniejącą treść. Ustawia wartość parametru **msg format** na MQFMT_TRIGGER.

Więcej szczegółów można znaleźć w opisie metody klasy ImqItem pod adresem [“Klasa ImqItem C++” na stronie 1970](#).

virtual ImqBoolean pasteIn (ImqMessage & komunikat);

Odczytuje strukturę danych MQTM z buforu komunikatów.

Aby możliwe było pomyślne działanie, ImqMessage **format** musi mieć wartość MQFMT_TRIGGER.

Więcej szczegółów można znaleźć w opisie metody klasy ImqItem pod adresem [“Klasa ImqItem C++” na stronie 1970](#).

Metody obiektów (publiczne)

void operator = (const ImqTrigger & wyzwalacz);

Kopiuje dane instancji z *wyzwalacza*, zastępując istniejące dane instancji.

ImqString applicationId () const ;

Zwraca kopię **identyfikatora aplikacji**.

void setApplicationId (const char * id);

Ustawia **identyfikator aplikacji**.

MQLONG applicationType () const ;

Zwraca **typ aplikacji**.

void setApplicationType (const MQLONG typ);

Ustawia **typ aplikacji**.

ImqBoolean copyOut (MQTMC2 * ptmc2);

Hermetyzuje strukturę danych MQTM, która jest otrzymana w kolejkach inicjuj. Wypełnia równoważną strukturę danych MQTMC2 udostępnianej przez program wywołujący i ustawia pole QMgrName (które nie znajduje się w strukturze danych MQTM) na wszystkie odstępy. Struktura danych MQTMC2 jest

tradycyjnie używana jako parametr dla aplikacji uruchamianych przez monitor wyzwalacza. Ta metoda zwraca wartość PRAWDA, jeśli powiodła się.

ImqString environmentData () const ;

Zwraca kopię **danych środowiska**.

void setEnvironmentData (const char * data);

Ustawia **dane środowiska**.

ImqString processName () const ;

Zwraca kopię **nazwy procesu**.

void setProcessName (const char * nazwa);

Ustawia **nazwę procesu**, dopełnianą spacjami do 48 znaków.

ImqString queueName () const ;

Zwraca kopię **nazwy kolejki**.

void setQueueName (const char * nazwa);

Ustawia **nazwę kolejki**, dopełniając odstępy do 48 znaków.

ImqString triggerData () const ;

Zwraca kopię **danych wyzwalacza**.

void setTriggerData (const char * dane);

Ustawia **dane wyzwalacza**.

ImqString userData () const ;

Zwraca kopię **danych użytkownika**.

void setUserData (const char * data);

Ustawia **dane użytkownika**.

Dane obiektu (chronione)

MQTM omqtm

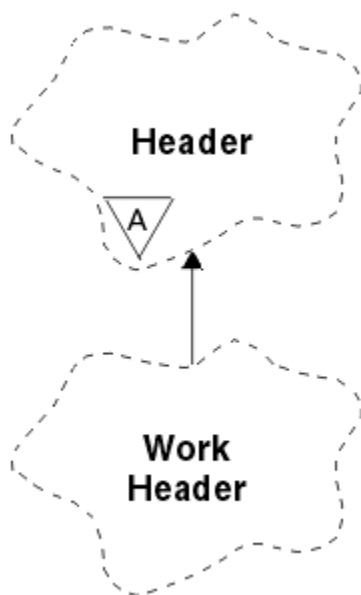
Struktura danych MQTM.

Kody przyczyny

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- Błąd MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_BŁĄD

Klasa ImqWorknagłówka C++

Ta klasa hermetyzuje określone funkcje struktury danych MQWIH.



Rysunek 69. Klasa nagłówka *ImqWork*

Obiekty tej klasy są używane przez aplikacje wstawiające komunikaty do kolejki zarządzanej przez program z/OS Workload Manager.

- [“Atrybuty obiektu” na stronie 2031](#)
- [“Konstruktory” na stronie 2031](#)
- [“Przeciążone metody ImqItem” na stronie 2031](#)
- [“Metody obiektów \(publiczne\)” na stronie 2032](#)
- [“Dane obiektu \(chronione\)” na stronie 2032](#)
- [“Kody przyczyny” na stronie 2032](#)

Atrybuty obiektu

znacznik komunikatu

Znacznik komunikatu dla programu z/OS Workload Manager o długości MQ_MSG_TOKEN_LENGTH. Wartością początkową jest MQMTOK_NONE.

nazwa usługi

32-znakowa nazwa procesu. Nazwa jest początkowo pusta.

krok usługi

8-znakowa nazwa kroku w procesie. Nazwa jest początkowo pusta.

Konstruktory

ImqWorkNagłówek ();

Konstruktor domyślny.

Nagłówek ImqWork(const ImqWorkHeader & header);

Konstruktor kopiowania.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut(ImqMessage & komunikat);

Wstawia strukturę danych MQWIH na początek buforu komunikatów, dalej przesuwając istniejące dane komunikatu, a następnie ustawia wartość **msg format** na wartość MQFMT_WORK_INFO_HEADER.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

virtual ImqBoolean pasteIn(ImqMessage & komunikat);

Odczytuje strukturę danych MQWIH z buforu komunikatów.

Aby możliwe było pomyślne, kodowanie obiektu *msg* musi mieć wartość MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT na wartość MQENC_NATIVE.

Format ImqMessage musi mieć wartość MQFMT_WORK_INFO_HEADER.

Więcej szczegółów można znaleźć w opisie metody klasy macierzystej.

Metody obiektów (publiczne)**void operator = (const ImqWorkHeader & header);**

Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqBinary messageToken () const;

Zwraca **token komunikatu**.

ImqBoolean setMessageToken (const ImqBinary & token);

Ustawia **token komunikatu**. Długość danych *token* musi mieć wartość zero lub wartość MQ_MSG_TOKEN_LENGTH. Zwraca wartość PRAWDA, jeśli powiodła się.

void setMessageToken (const MQBYTE16 token = 0);

Ustawia **token komunikatu**. *token* może być zerem, który jest taki sam, jak parametr MQMTOK_NONE. Jeśli element *token* ma wartość niezerową, musi być adresowany do bajtów MQ_MSG_TOKEN_LENGTH bajtów danych binarnych.

W przypadku korzystania z predefiniowanych wartości, takich jak MQMTOK_NONE, może być konieczne użycie rzutowania w celu zapewnienia zgodności podpisu, na przykład (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () const;

Zwraca **nazwę usługi**, w tym odstępy końcowe.

void setServiceName (const char * nazwa);

Ustawia **nazwę usługi**.

ImqString serviceStep () const;

Zwraca **krok usługi**, w tym końcowe odstępy.

void setServiceStep (const char * krok);

Ustawia **krok usługi**.

Dane obiektu (chronione)**MQWIH omqwih**

Struktura danych MQWIH.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Właściwości obiektów IBM MQ classes for JMS

Wszystkie obiekty w programie IBM MQ classes for JMS mają właściwości. Różne właściwości mają zastosowanie do różnych typów obiektów. Różne właściwości mają różne dozwolone wartości, a wartości właściwości symbolicznych różnią się między narzędziem administracyjnym a kodem programu.

Produkt IBM MQ classes for JMS udostępnia narzędzia do ustawiania i wykonywania zapytań dotyczących właściwości obiektów przy użyciu narzędzia administracyjnego IBM MQ JMS, programu IBM MQ Explorer lub aplikacji. Wiele właściwości ma znaczenie tylko dla konkretnego podzbioru typów obiektów.

Informacje na temat korzystania z narzędzia administracyjnego IBM MQ JMS zawiera sekcja [Konfigurowanie obiektów produktu JMS przy użyciu narzędzia administracyjnego](#).

Program [Tabela 226](#) na stronie 2033 zawiera krótki opis każdej właściwości i wyświetlane dla każdej właściwości typy obiektów, do których ma zastosowanie. Typy obiektów są identyfikowane za pomocą słów kluczowych. W celu wyjaśnienia tych obiektów należy zapoznać się z informacjami znajdującymi się w sekcji [Konfigurowanie obiektów produktu JMS przy użyciu narzędzia administracyjnego](#).

Liczby odnoszą się do uwag na końcu tabeli. Patrz także [“Zależności między właściwościami obiektów produktu IBM MQ classes for JMS”](#) na stronie 2036.

Właściwość składa się z pary nazwa-wartość w formacie:

```
PROPERTY_NAME(property_value)
```

Tematy na tej liście sekcji, dla każdej właściwości, nazwy właściwości i krótkiego opisu, a także wyświetlane są poprawne wartości właściwości używane w narzędziu administracyjnym. i metoda set, która jest używana do ustawiania wartości właściwości w aplikacji. W tematach przedstawiono także poprawne wartości właściwości dla każdej właściwości oraz odwzorowanie między wartościami właściwości symbolicznych używanych w narzędziu a ich programowalnymi odpowiednikami.

W nazwach właściwości nie jest rozróżniana wielkość liter i są one ograniczone do zbioru rozpoznanych nazw w tych tematach.

<i>Tabela 226. Nazwy właściwości i mające zastosowanie typy obiektów</i>									
Właściwość	Postać krótka	Typ obiektu							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“APPLICATIONNAME” na stronie 2038	APPNAME	Y	Y	Y			Y	Y	Y
“WYJĄTEK ASYNCEXCEPTION” na stronie 2038	AEX	Y	Y	Y			Y	Y	Y
“BROKERCCDURSUBQ” na stronie 2039 ¹	CCDSUB					Y			
“BROKERCCSUBQ” na stronie 2040 ¹	CCSUB	Y		Y			Y		Y
“BROKERCONQ” na stronie 2040 ¹	BCON	Y		Y			Y		Y
“BROKERDURSUBQ” na stronie 2041 ¹	BDSUB					Y			
“BROKERPUBQ” na stronie 2041 ¹	BPUB	Y		Y		Y	Y		Y
“BROKERPUBQMGR” na stronie 2042 ¹	BPQM					Y			
“BROKERQMGR” na stronie 2042 ¹	BQM	Y		Y			Y		Y
“BROKERSUBQ” na stronie 2042 ¹	BSUB	Y		Y			Y		Y
“BROKERVER” na stronie 2043 ¹	BVER	Y ²		Y ²		Y	Y		Y
“CCDTURL” na stronie 2044 ³	CCDT	Y	Y	Y			Y	Y	Y
“CCSID” na stronie 2044	CCS	Y	Y	Y	Y	Y	Y	Y	Y
“CHANNEL” na stronie 2044 ³	CHAN	Y	Y	Y			Y	Y	Y
“CLEANUP” na stronie 2045 ¹	CL	Y		Y			Y		Y
“CLEANUPINT” na stronie 2045 ¹	CLINT	Y		Y			Y		Y
“Lista CONNECTIONNAMELIST” na stronie 2046	CNLIST	Y	Y	Y					

Tabela 226. Nazwy właściwości i mające zastosowanie typy obiektów (kontynuacja)

Właściwość	Postać krótka	Typ obiektu							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“CLIENTRECONNECTOPTIONS” na stronie 2046	CROPT	Y	Y	Y					
“CLIENTRECONNECTTIMEOUT” na stronie 2047	CRT	Y	Y	Y					
“CLIENTID” na stronie 2048	CID	Y ²	Y	Y ²			Y	Y	Y
“CLONESUPP” na stronie 2048	CLS	Y		Y			Y		Y
“COMPHDR” na stronie 2049	HC	Y		Y			Y		Y
“COMPMSG” na stronie 2049	MC	Y	Y	Y			Y	Y	Y
“CONNOPT” na stronie 2049	CNOPT	Y	Y	Y			Y	Y	Y
“CONNTAG” na stronie 2051	CNTAG	Y	Y	Y			Y	Y	Y
“opis” na stronie 2051	DESC	Y ²	Y	Y ²	Y	Y	Y	Y	Y
“DIRECTAUTH” na stronie 2051	DAUTH	Y ²		Y ²					
“ENCODING” na stronie 2052	ENC				Y	Y			
“EXPIRY” na stronie 2053	EXP				Y	Y			
“FAILIFQUIESCE” na stronie 2053	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
“HOSTNAME” na stronie 2054	HOST	Y ²	Y	Y ²			Y	Y	Y
“LOCALADDRESS” na stronie 2055	LA	Y ²	Y	Y ²			Y	Y	Y
“MAPNAMESTYLE” na stronie 2055	MNST	Y	Y	Y			Y	Y	Y
“MAXBUFFSIZE” na stronie 2056	MBSZ	Y ²		Y ²					
“MDREAD” na stronie 2056	MDR				Y	Y			
“MDWRITE” na stronie 2057	MDW				Y	Y			
“MDMSGCTX” na stronie 2057	MDCTX				Y	Y			
“MSGBATCHSZ” na stronie 2058 ¹	MBS	Y	Y	Y			Y	Y	Y
“MSGBODY” na stronie 2059	MBODY				Y	Y			
“MSGRETENTION” na stronie 2059	MRET	Y	Y				Y	Y	
“MSGSELECTION” na stronie 2060 ¹	MSEL	Y		Y			Y		Y
“MULTICAST” na stronie 2060	MCAST	Y ²		Y ²		Y			
“OPTIMISTICPUBLICATION” na stronie 2061 ¹	OPTPUB	Y		Y					
“OUTCOMENOTIFICATION” na stronie 2062 ¹	NOTIFY	Y		Y					
“PERSISTENCE” na stronie 2062	PER				Y	Y			
“POLLINGINT” na stronie 2063 ¹	PINT	Y	Y	Y			Y	Y	Y
“PORT” na stronie 2063	PORT	Y ²	Y	Y ²			Y	Y	Y
“PRIORYTET” na stronie 2064	PRI				Y	Y			

Tabela 226. Nazwy właściwości i mające zastosowanie typy obiektów (kontynuacja)

Właściwość	Postać krótka	Typ obiektu							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“PROCESSDURATION” na stronie 2064 ¹	PROC DUR	Y		Y					
“PROVIDERVERSION” na stronie 2065	PVER	Y	Y	Y			Y	Y	Y
“PROXYHOSTNAME” na stronie 2067	PHOST	Y ²		Y ²					
“PROXYPORT” na stronie 2068	PPORT	Y ²		Y ²					
“PUBACKINT” na stronie 2068 ¹	PAI	Y		Y			Y		Y
“PUTASYNCAALLOWED” na stronie 2069	PAALD				Y	Y			
“QMANAGER” na stronie 2069	QMGR	Y	Y	Y	Y		Y	Y	Y
“QUEUE” na stronie 2070	QU				Y				
“READAHEADALLOWED” na stronie 2070	RAALD				Y	Y			
“READAHEADCLOSEPOLICY” na stronie 2071	RACP				Y	Y			
“RECEIVECCSID” na stronie 2071	RCCS				Y	Y			
“RECEIVECONVERSION” na stronie 2072	RCNV				Y	Y			
“RECEIVEISOLATION” na stronie 2072 ¹	RCVISOL	Y		Y					
“RECEXIT” na stronie 2073	RCX	Y	Y	Y			Y	Y	Y
“RECEXITINIT” na stronie 2073	RCXI	Y	Y	Y			Y	Y	Y
“REPLYTOSTYLE” na stronie 2074	RTOST				Y	Y			
“RESCANINT” na stronie 2074 ¹	RINT	Y	Y				Y	Y	
“SECEXIT” na stronie 2075	SCX	Y	Y	Y			Y	Y	Y
“SECEXITINIT” na stronie 2076	SCXI	Y	Y	Y			Y	Y	Y
“SENDCHECKCOUNT” na stronie 2076	SCC	Y	Y	Y			Y	Y	Y
“SENDEXIT” na stronie 2076	SDX	Y	Y	Y			Y	Y	Y
“SENDEXITINIT” na stronie 2077	SDXI	Y	Y	Y			Y	Y	Y
“SHARECONVALLOWED” na stronie 2078	SCALD	Y	Y	Y			Y	Y	Y
“SPARSESUBS” na stronie 2078 ¹	SSUBS	Y		Y					
“SSLCIPHERSUITE” na stronie 2079	SCPHS	Y	Y	Y			Y	Y	Y
“SSLCRL” na stronie 2079	SCRL	Y	Y	Y			Y	Y	Y
“SSLFIPSREQUIRED” na stronie 2079	SFIPS	Y	Y	Y			Y	Y	Y
“SSLPEERNAME” na stronie 2080	SPEER	Y	Y	Y			Y	Y	Y

Tabela 226. Nazwy właściwości i mające zastosowanie typy obiektów (kontynuacja)

Właściwość	Postać krótka	Typ obiektu							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
“SSLRESETCOUNT” na stronie 2080	SRC	Y	Y	Y			Y	Y	Y
“STATREFRESHINT” na stronie 2081 ¹	SRI	Y		Y			Y		Y
“SUBSTORE” na stronie 2081 ¹	SS	Y		Y			Y		Y
“SYNCPOINTALLGETS” na stronie 2082	SPAG	Y	Y	Y			Y	Y	Y
“TARGCLIENT” na stronie 2082	TC				Y	Y			
“TARGCLIENTMATCHING” na stronie 2083	TCM	Y	Y				Y	Y	
“TEMPMODEL” na stronie 2083	TM	Y	Y				Y	Y	
“TEMPQPREFIX” na stronie 2084	TQP	Y	Y				Y	Y	
“TEMPTOPICPREFIX” na stronie 2084	TTP	Y		Y			Y		Y
“TOPIC” na stronie 2085	TOP					Y			
“TRANSPORT” na stronie 2085	TRAN	Y ²	Y	Y ²			Y	Y	Y
“WILDCARDFORMAT” na stronie 2086	WCFMT	Y		Y			Y		Y

Uwaga:

1. Ta właściwość może być używana z produktem Wersja 7.0 w produkcie IBM MQ classes for JMS , ale nie ma wpływu na aplikację połączoną z menedżerem kolejek produktu Wersja 7.0 , chyba że właściwość PROVIDERVERSION fabryki połączeń jest ustawiona na numer wersji mniejszy niż 7.
2. Tylko właściwości BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT i TRANSPORT są obsługiwane dla obiektu fabryki ConnectionFactory lub TopicConnection podczas korzystania z połączenia w czasie rzeczywistym z brokerem.
3. Właściwości CCDURL i CHANNEL obiektu nie mogą być jednocześnie ustawione jednocześnie.

Zależności między właściwościami obiektów produktu IBM MQ classes for JMS

Poprawność niektórych właściwości jest zależna od konkretnych wartości innych właściwości.

Zależność ta może wystąpić w następujących grupach właściwości:

- Właściwości klienta
- Właściwości połączenia w czasie rzeczywistym z brokerem
- Wyjdz z łańcuchów inicjowania

Właściwości klienta

W przypadku połączenia z menedżerem kolejek następujące właściwości są istotne tylko wtedy, gdy parametr TRANSPORT ma wartość CLIENT:

- HOSTNAME
- PORT
- CHANNEL

- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- REEXIT
- REEXITINIT
- SEEXIT
- SEEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

Nie można ustawić wartości dla tych właściwości przy użyciu narzędzia administracyjnego, jeśli TRANSPORT ma wartość BIND.

Jeśli TRANSPORT ma wartość CLIENT, domyślną wartością właściwości BROKERVER jest V1 , a wartością domyślną właściwości PORT jest 1414. Jeśli wartość parametru BROKERVER lub PORT zostanie jawnie ustawiona, późniejsza zmiana wartości TRANSPORT nie spowoduje nadpisania wybranych opcji.

Właściwości połączenia w czasie rzeczywistym z brokerem

Jeśli parametr TRANSPORT ma wartość DIRECT lub DIRECTHTTP, istotne są tylko następujące właściwości:

- BROKERVER
- CLIENTID
- opis
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (obsługiwane tylko dla DIRECT)
- PORT
- PROXYHOSTNAME (obsługiwane tylko dla DIRECT)
- PROXYPORT (obsługiwany tylko dla DIRECT)

Jeśli TRANSPORT ma wartość DIRECT lub DIRECTHTTP, wartością domyślną właściwości BROKERVER jest V2, a domyślną wartością właściwości PORT jest 1506. Jeśli wartość parametru BROKERVER lub PORT zostanie jawnie ustawiona, późniejsza zmiana wartości TRANSPORT nie spowoduje nadpisania wybranych opcji.

Wyjdz z łańcuchów inicjowania

Nie należy ustawiać żadnego z łańcuchów inicjowania wyjścia bez podawania odpowiedniej nazwy wyjścia. Właściwości inicjowania wyjścia to:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Na przykład określenie RECEXITINIT(myString) bez określania RECEXIT(some.exit.classname) powoduje wystąpienie błędu.

Odsyłacze pokrewne

[“TRANSPORT” na stronie 2085](#)

Rodzaj połączenia z menedżerem kolejek lub brokerem.

APPLICATIONNAME

Aplikacja może ustawić nazwę, która identyfikuje jego połączenie z menedżerem kolejek. Ta nazwa aplikacji jest wyświetlana za pomocą komendy **DISPLAY CONN MQSC/PCF** (gdzie pole to jest nazywane **APPLTAG**) lub na ekranie **Połączenia aplikacji** programu IBM MQ Explorer (gdzie pole to jest nazywane **App name**).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : APPLICATIONNAME

Krótką nazwą narzędzia administracyjnego JMS : APPNAME

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setAppNazwa ()
- MQConnectionFactory.getAppNazwa ()

Wartości

Dowolny poprawny łańcuch, który nie może być dłuższy niż 28 znaków. Dłuższe nazwy są dopasowywane tak, aby zmieściły się, usuwając początkowe nazwy pakietów, jeśli jest to konieczne. Na przykład, jeśli klasą wywołującym jest com.example.MainApp, używana jest pełna nazwa, ale jeśli klasą wywołującym jest com.example.dictionaryAndThesaurus.multilingual.mainApp, używana jest nazwa multilingual.mainApp, ponieważ jest to najdłuższa kombinacja nazwy klasy i najbardziej należyj nazwy pakietu, która mieści się w dostępnej długości.

Jeśli sama nazwa klasy ma więcej niż 28 znaków, zostanie obcięta do dopasowania. Na przykład com.example.mainApplicationForSecondTestCase staje się mainApplicationForSecondTest.

WYJĄTEK ASYNCEXCEPTION

Ta właściwość określa, czy program IBM MQ classes for JMS informuje obiekt ExceptionListener tylko wtedy, gdy połączenie jest zerwane, lub gdy dowolny wyjątek występuje asynchronicznie w wywołaniu funkcji API produktu JMS. Dotyczy to wszystkich połączeń utworzonych z tej fabryki połączeń ConnectionFactory, dla których zarejestrowano obiekt ExceptionListener.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : ASYNCEXCEPTION

Krótką nazwa narzędzia administracyjnego JMS : AEX

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setAsyncWyjątki ()`
- `MQConnectionFactory.getAsyncWyjątki ()`

Wartości

ASYNC_EXCEPTIONS_ALL

Wszystkie wyjątki wykryte asynchronicznie, poza zasięgiem wywołania synchronicznego interfejsu API, oraz wszystkie wyjątki zerwane połączenia są wysyłane do obiektu `ExceptionListener`.

Środowisko	Wartość
JMS narzędzie administracyjne	ALL
Programowe	<code>WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1</code>
IBM MQ Eksplorator	Wszystkie

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Do obiektu `ExceptionListener` wysyłane są tylko wyjątki wskazujące, że połączenie zerwane jest zerwane. Wszystkie inne wyjątki występujące podczas przetwarzania asynchronicznego nie są raportowane do obiektu `ExceptionListener` dlatego aplikacja nie jest powiadamiana o tych wyjątkach.

V 8.0.0.2 Jest to wartość domyślna z produktu IBM MQ 8.0.0, Fix Pack 2 (patrz sekcja [JMS: Zmiany procesu następowania wyjątków w wersji 8.0](#)).

Środowisko	Wartość
JMS narzędzie administracyjne	POŁĄCZONO
Programowe	<code>WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1</code>
MQ Explorer	Zerwane połączenie

Zdefiniowana jest następująca stała dodatkowa:

- **V 8.0.0.2** W systemie Wersja 8.0.0, pakiet poprawek 2:
`WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN`
- Przed Wersja 8.0.0, pakiet poprawek 2: `WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL`

Informacje pokrewne

[Wyjątki w produkcie IBM MQ classes for JMS](#)

BROKERCCDURSUBQ

Nazwa kolejki, z której pobierane są komunikaty trwałej subskrypcji dla obiektu `ConnectionConsumer`.

Obiekty mające zastosowanie

Temat

Długa nazwa narzędzia administracyjnego JMS : `BROKERCCDURSUBQ`

Krótką nazwa narzędzia administracyjnego JMS : CCDSUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Wartości

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERCCSUBQ

Nazwa kolejki, z której pobierane są nietrwałe komunikaty subskrypcji dla obiektu ConnectionConsumer.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERCCSUBQ

Krótką nazwa narzędzia administracyjnego JMS : CCSUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Wartości

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERCONQ

Nazwa kolejki sterującej brokera.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERCONQ

Krótką nazwa narzędzia administracyjnego JMS : BCON

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Wartości

SYSTEM.BROKER.CONTROL.QUEUE

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERDURSUBQ

Gdy produkt IBM MQ classes for JMS jest używany w trybie migracji dostawcy przesyłania komunikatów produktu IBM MQ , ta właściwość określa nazwę kolejki, z której pobierane są komunikaty trwałej subskrypcji.

Obiekty mające zastosowanie

Temat

Długa nazwa narzędzia administracyjnego JMS : BROKERDURSUBQ

Krótką nazwa narzędzia administracyjnego JMS : BDSUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Wartości

SYSTEM.JMS.D.SUBSCRIBER.QUEUE

Jest to wartość domyślna.

Dowolny poprawny łańcuch

Zaczynając od SYSTEM.JMS.D

Informacje pokrewne

[Reguły wybierania trybu dostawcy przesyłania komunikatów produktu IBM MQ](#)

BROKERPUBQ

Nazwa kolejki, w której są wysyłane opublikowane komunikaty (kolejka strumienia).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, Topic, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERPUBQ

Krótką nazwa narzędzia administracyjnego JMS : BPUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Wartości

SYSTEM.BROKER.DEFAULT.STREAM

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERPUBQMGR

Nazwa menedżera kolejek, który jest właścicielem kolejki, do której wysyłane są komunikaty opublikowane w tym temacie.

Obiekty mające zastosowanie

Temat

Długa nazwa narzędzia administracyjnego JMS : BROKERPUBQMGR

Krótką nazwa narzędzia administracyjnego JMS : BPQM

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Wartości

null

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERQMGR

Nazwa menedżera kolejek, w którym działa broker.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERQMGR

Krótką nazwa narzędzia administracyjnego JMS : BQM

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Wartości

null

Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERSUBQ

Gdy produkt IBM MQ classes for JMS jest używany w trybie migracji dostawcy przesyłania komunikatów produktu IBM MQ , ta właściwość określa nazwę kolejki, z której pobierane są nietrwałe komunikaty subskrypcji.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERSUBQ

Krótką nazwa narzędzia administracyjnego JMS : BSUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Wartości

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

Jest to wartość domyślna.

Dowolny poprawny łańcuch

Zaczynając od SYSTEM.JMS.ND

Informacje pokrewne

[Reguły wybierania trybu dostawcy przesyłania komunikatów produktu IBM MQ](#)

BROKERVER

Wersja używanego brokera.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, Topic, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : BROKERVER

Krótką nazwa narzędzia administracyjnego JMS : BVER

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setBrokerWersja ()
- MQConnectionFactory.getBrokerWersja ()

Wartości

V1

Aby użyć brokera publikowania/subskrypcji produktu IBM MQ lub brokera produktu IBM MQ Integrator, WebSphere Event Broker, produktu WebSphere Business Integration Event Broker lub produktu WebSphere Business Integration Message Broker w trybie zgodności. Jest to wartość domyślna, jeśli TRANSPORT jest ustawiony na wartość BIND lub CLIENT.

V2

Aby korzystać z brokera produktu IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie rodzimym. Jest to wartość domyślna, jeśli TRANSPORT jest ustawiony na wartość DIRECT lub DIRECTHTTP.

nieokreślona

Po przeprowadzeniu migracji brokera z wersji V6 do wersji V7 ustaw tę właściwość w taki sposób, aby nagłówki RFH2 nie były już używane. Po migracji ta właściwość nie jest już istotna.

CCDTURL

Adres URL (Uniform Resource Locator), który identyfikuje nazwę i położenie pliku zawierającego tabelę definicji kanału klienta i określa sposób uzyskiwania dostępu do tego pliku.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CCDTURL

Krótką nazwa narzędzia administracyjnego JMS : CCDT

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Wartości

null

Jest to wartość domyślna.

Adres URL (Uniform Resource Locator)

CCSID

Identyfikator kodowanego zestawu znaków, który ma być używany dla połączenia lub miejsca docelowego.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, kolejka, temat, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CCSID

Krótką nazwa narzędzia administracyjnego JMS : CCS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Wartości

819

Jest to wartość domyślna dla fabryki połączeń.

1208

Jest to wartość domyślna dla miejsca docelowego.

Dowolna dodatnia liczba całkowita

CHANNEL

Nazwa kanału połączenia klienckiego, który jest używany.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CHANNEL

Krótką nazwa narzędzia administracyjnego JMS : CHAN

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Wartości

SYSTEM.DEF.SVRCONN

Jest to wartość domyślna.

Dowolny poprawny łańcuch

CLEANUP

Poziom procedury czyszczącej dla składnic subskrypcji BROKER lub MIGRATE.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CLEANUP

Krótką nazwa narzędzia administracyjnego JMS : CL

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setCleanupLevel ()
- MQConnectionFactory.getCleanupPoziom ()

Wartości

Bezpieczne

Użyj bezpiecznej procedury czyszczącej. Jest to wartość domyślna.

ASPROP

Należy używać bezpiecznego, mocnego lub bez czyszczenia zgodnie z właściwością ustawioną w wierszu komend produktu Java .

BRAK

Nie używaj procedury czyszczącej.

silny

Użyj silnego czyszczenia.

CLEANUPINT

Odstęp czasu (w milisekundach) między kolejnymi uruchomieniami w tle programu narzędziowego do czyszczenia publikowania/subskrypcji.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CLEANUPINT

Krótką nazwa narzędzia administracyjnego JMS : CLINT

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setCleanupInterwał ()
- MQConnectionFactory.getCleanupInterwał ()

Wartości

3600000

Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

Lista CONNECTIONNAMESLIST

Lista nazw połączeń TCP/IP. Lista jest podejmowana w kolejności, raz na każdą próbę ponownego nawiązania połączenia.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : CONNECTIONNAMESLIST

Krótką nazwa narzędzia administracyjnego JMS : CNLIST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setconnectionNameList ()
- MQConnectionFactory.getconnectionNameList ()

Wartości

Rozdzielana przecinkami lista HOSTNAME (PORT). Parametr HOSTNAME może być nazwą DNS lub adresem IP.

Domyślny PORT to 1414.

CLIENTRECONNECTOPTIONS

Opcje regulujące ponowne połączenie.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLIENTRECONNECTOPTIONS

Krótką nazwa narzędzia administracyjnego JMS : CROPT

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setClientReconnectOptions()`
- `MQConnectionFactory.getClientReconnectOptions()`

Wartości

QMGR

Aplikacja może ponownie nawiązać połączenie, jednak tylko z menedżerem kolejek, z którym wcześniej nawiązane było połączenie.

Tej wartości należy użyć, jeśli aplikacja może zostać ponownie połączona, ale istnieje powinowactwo między aplikacją IBM MQ classes for JMS a menedżerem kolejek, do którego najpierw nawiązało połączenie.

Tę wartość należy wybrać, jeśli aplikacja ma automatycznie ponownie łączyć się z instancją rezerwową menedżera kolejek o wysokiej dostępności.

Aby użyć tej wartości programowo, należy użyć stałej `WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR`.

ANY

Aplikacja może ponownie nawiązać połączenie z dowolnym menedżerem kolejek.

Opcji ponownego połączenia należy używać tylko wtedy, gdy nie ma powinowactwa między klasami produktu IBM MQ dla aplikacji JMS a menedżerem kolejek, z którym początkowo nawiązało połączenie.

Aby użyć tej wartości z programu, należy użyć stałej `WMQConstants.WMQ_CLIENT_RECONNECT`.

WYŁĄCZONE

Połączenie aplikacji nie będzie ponownie nawiązywane.

Aby użyć tej wartości programowo, należy użyć stałej `WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED`.

ASDEF

To, czy aplikacja będzie ponownie łączyć się automatycznie, zależy od wartości atrybutu kanału IBM MQ `DefReconnect`.

Aby użyć tej wartości z programu, należy użyć stałej `WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF`.

CLIENTRECONNECTTIMEOUT

Czas przed zakończeniem ponownych prób ponownego nawiązania połączenia.

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `QueueConnection`, fabryka `TopicConnection`

Długa nazwa narzędzia administracyjnego JMS : `CLIENTRECONNECTTIMEOUT`

Krótką nazwa narzędzia administracyjnego JMS : `CRT`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setClientReconnectTimeout()`
- `MQConnectionFactory.getClientReconnectTimeout()`

Wartości

Odstęp czasu w sekundach. Domyślna wartość 1800 (30 minut).

CLIENTID

Identyfikator klienta służy do jednoznacznej identyfikacji połączenia aplikacji dla subskrypcji statycznych.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CLIENTID

Krótką nazwa narzędzia administracyjnego JMS : CID

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setClientId ()
- MQConnectionFactory.getClientId ()

Wartości

null

Jest to wartość domyślna.

Dowolny poprawny łańcuch

CLONESUPP

Określa, czy dwie lub więcej instancji tego samego, trwałego subskrybenta tematów może być uruchomione jednocześnie.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CLONESUPP

Krótką nazwa narzędzia administracyjnego JMS : CLS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setCloneSupport ()
- MQConnectionFactory.getClone-obstuga ()

Wartości

WYŁĄCZONE

W danym momencie może być uruchomiona tylko jedna instancja trwałego subskrybenta tematów.
Jest to wartość domyślna.

WŁĄCZONY

Co najmniej dwie instancje tego samego subskrybenta trwałego tematu mogą być uruchamiane jednocześnie, ale każda instancja musi być uruchamiana na osobnej maszynie wirtualnej Java (JVM).

COMPHDR

Lista technik, które mogą być używane do kompresowania danych nagłówka w połączeniu.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : COMPHDR

Krótką nazwa narzędzia administracyjnego JMS : HC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Wartości

BRAK

Jest to wartość domyślna.

SYSTEM

Wykonywana jest kompresja nagłówka komunikatu RLE.

COMPMSG

Lista technik, które mogą być używane do kompresowania danych komunikatu w połączeniu.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : COMPMSG

Krótką nazwa narzędzia administracyjnego JMS : MC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Wartości

BRAK

Jest to wartość domyślna.

Lista co najmniej jednej z następujących wartości oddzielonych odstępami:

RLE ZLIBFAST ZLIBHIGH

CONNOPT

Określa sposób, w jaki aplikacje produktu IBM MQ classes for JMS , które korzystają z transportu powiązań, łączą się z menedżerem kolejek.

Obiekty mające zastosowanie

Fabryki połączeń `ConnectionFactory`, `QueueConnection`, fabryka `TopicConnection`, fabryka `XAConnectionFactory`, fabryka `XAQueueConnection`, fabryka `XATopicConnection`.

Długa nazwa narzędzia administracyjnego JMS : CONNOPT

Krótką nazwa narzędzia administracyjnego JMS : CNOPT

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.OpcjesetMQConnection()`
- `MQConnectionFactory.OpcjegetMQConnection()`

Wartości

STANDARDOWA

Rodzaj powiązania między aplikacją a menedżerem kolejek zależy od wartości atrybutu *DefaultBind(Typ powiązania)* menedżera kolejek. Wartość STANDARD jest odwzorowana na wartość parametru IBM MQ *ConnectOption* `MQCNO_STANDARD_BINDING`.

Współużytkowane

Aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania, ale współużytkują niektóre zasoby. Ta wartość jest odwzorowana na wartość IBM MQ *ConnectOption* `MQCNO_SHARED_BINDING`.

Odizolowane

Aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania i nie współużytkują żadnych zasobów. Wartość ISOLATED jest odwzorowywane na wartość IBM MQ *ConnectOption* `MQCNO_ISOLATED_BINDING`.

Krótką ścieżką

Aplikacja i agent lokalnego menedżera kolejek są uruchamiane w tej samej jednostce wykonywania. Ta wartość jest odwzorowywana na powiązanie z produktem IBM MQ *ConnectOption* `MQCNO_FASTPATH_BINDING`.

SERIALQM

Aplikacja żąda wyłącznego użycia znacznika połączenia w zasięgu menedżera kolejek. Ta wartość jest odwzorowana na wartość IBM MQ *ConnectOption* `MQCNO_SERIALIZE_CONN_TAG_Q_MGR`.

SERIALQSG

Aplikacja żąda wyłącznego użycia znacznika połączenia w zasięgu grupy współużytkowania kolejki, do której należy menedżer kolejek. Wartość SERIALQSG odwzorowuje się na wartość IBM MQ *ConnectOption* `MQCNO_SERIALIZE_CONN_TAG_QSG`.

OGRANICZENIETQM

Aplikacja żąda współużytkowanego użycia znacznika połączenia, ale istnieją ograniczenia dotyczące współużytkowania współużytkowanego znacznika połączenia w zasięgu menedżera kolejek. Ta wartość jest odwzorowywana na wartość IBM MQ *ConnectOption* `MQCNO_RESTRICT_CONN_TAG_Q_MGR`.

OGRANICZONAQSG

Aplikacja żąda współużytkowanego użycia znacznika połączenia, ale istnieją ograniczenia dotyczące współużytkowania współużytkowanego znacznika połączenia w zasięgu grupy współużytkowania kolejek, do której należy menedżer kolejek. Ta wartość jest odwzorowywana na wartość IBM MQ *ConnectOption* `MQCNO_RESTRICT_CONN_TAG_QSG`.

Więcej informacji na temat opcji połączenia z produktem IBM MQ zawiera sekcja [Nawiązywanie połączenia z menedżerem kolejek przy użyciu wywołania MQCONN](#).

CONNTAG

Znacznik, który menedżer kolejek wiąże z zasobami zaktualizowanymi przez aplikację w ramach jednostki pracy, gdy aplikacja jest połączona z menedżerem kolejek.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CONNTAG

Krótką nazwa narzędzia administracyjnego JMS : CNTAG

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setConnZnacznik ()
- MQConnectionFactory.getConnZnacznik ()

Wartości

Tablica bajtów o długości 128 elementów, w której każdy element ma wartość 0

Jest to wartość domyślna.

Dowolny łańcuch

Wartość jest obcinana, jeśli jest dłuższa niż 128 bajtów.

opis

Opis składowanego obiektu.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, kolejka, temat, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : OPIS

Krótką nazwa narzędzia administracyjnego JMS : DESC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Wartości

null

Jest to wartość domyślna.

Dowolny poprawny łańcuch

DIRECTAUTH

Określa, czy uwierzytelnianie SSL jest używane w czasie rzeczywistym do połączenia z brokerem.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : DIRECTAUTH

Krótką nazwa narzędzia administracyjnego JMS : DAUTH

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory.getDirectAuth ()

Wartości

BASIC

Brak uwierzytelniania, uwierzytelnianie za pomocą nazwy użytkownika lub uwierzytelnianie za pomocą hasła. Jest to wartość domyślna.

Certyfikat

Uwierzytelnianie certyfikatu klucza publicznego.

ENCODING

Sposób, w jaki dane liczbowe w treści komunikatu są reprezentowane, gdy komunikat jest wysyłany do tego miejsca docelowego. Ta właściwość określa reprezentację binarnych liczb całkowitych, upakowanych liczb całkowitych dziesiętnych i liczb zmiennopozycyjnych.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : ENCODING

Krótką nazwa narzędzia administracyjnego JMS : ENC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Wartości

Właściwość ENCODING

Poprawne wartości, jakie może wykonać właściwość ENCODING , są skonstruowane z trzech podwłaściwości:

Kodowanie na podstawie liczb całkowitych

Normalny lub odwrócony

Kodowanie dziesiętne

Normalny lub odwrócony

Kodowanie zmiennopozycyjne

IEEE normal, IEEE reversed, or z/OS

Właściwość `ENCODING` jest wyrażona w postaci łańcucha o długości trzech znaków, z następującą składnią:

```
{N|R}{N|R}{N|R|3}
```

W tym łańcuchu:

- N oznacza normalny
- R oznacza odwrócone
- 3 oznacza z/OS
- Pierwszy znak reprezentuje *kodowanie liczb całkowitych* .
- Drugi znak reprezentuje *kodowanie dziesiętne* .
- Trzeci znak reprezentuje *kodowanie zmiennopozycyjne* .

Udostępnia zestaw dwunastu możliwych wartości dla właściwości `ENCODING` .

Istnieje dodatkowa wartość, łańcuch `NATIVE`, który ustawia odpowiednie wartości kodowania dla platformy Java .

W poniższych przykładach przedstawiono poprawne kombinacje dla produktu `ENCODING`:

```
ENCODING (NNR)  
ENCODING (NATIVE)  
ENCODING (RR3)
```

EXPIRY

Czas, po upływie którego komunikaty w miejscu docelowym tracą ważność.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : `WAŻNOŚCI`

Krótką nazwą narzędzia administracyjnego JMS : `EXP`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQDestination.setExpiry()`
- `MQDestination.getExpiry()`

Wartości

APP

Termin utraty ważności może być zdefiniowany przez aplikację JMS . Jest to wartość domyślna.

UNLIM

Nie występuje utrata ważności.

0

Nie występuje utrata ważności.

Dowolna dodatnia liczba całkowita reprezentująca utratę ważności w milisekundach.

FAILIFQUIESCE

Ta właściwość określa, czy wywołania do pewnych metod nie powiodą się, jeśli menedżer kolejek jest w stanie wygaszania, albo aplikacja nawiązuje połączenie z menedżerem kolejek przy użyciu transportu

CLIENT, a kanał używany przez aplikację został przełączony w stan wygaszania, na przykład za pomocą komendy MQSC **STOP CHANNEL** lub **STOP CHANNEL MODE(QUIESCE)** .

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, kolejka, temat, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : FAILIFQUIESCE

Krótką nazwa narzędzia administracyjnego JMS : FIQ

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Wartości

YES

Wywołania niektórych metod nie powiodą się, jeśli albo menedżer kolejek jest w stanie wygaszania, albo kanał używany do łączenia się z menedżerem kolejek jest wygaszany. Jeśli aplikacja wykryje którekolwiek z tych warunków, aplikacja może zakończyć swoje natychmiastowe zadanie i zamknąć połączenie, co umożliwi zatrzymanie menedżera kolejek lub instancji kanału. Jest to wartość domyślna.

NO

Wywołanie metody nie powiodło się, ponieważ menedżer kolejek lub kanał używany do łączenia się z menedżerem kolejek jest w stanie wygaszania. Jeśli zostanie określona ta wartość, aplikacja nie będzie mogła wykryć, że menedżer kolejek lub kanał jest wygaszany. Aplikacja może kontynuować wykonywanie operacji względem menedżera kolejek i w związku z tym zapobiec zatrzymaniu menedżera kolejek.

HOSTNAME

W przypadku połączenia z menedżerem kolejek: nazwa hosta lub adres IP systemu, na którym jest uruchomiony menedżer kolejek lub, w przypadku połączenia w czasie rzeczywistym z brokerem, nazwa hosta lub adres IP systemu, na którym jest uruchomiony broker.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : HOSTNAME

Krótką nazwa narzędzia administracyjnego JMS : HOST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setHostNazwa ()
- MQConnectionFactory.getHostNazwa ()

Wartości

localhost

Jest to wartość domyślna.

LOCALADDRESS

W przypadku połączenia z menedżerem kolejek ta właściwość określa albo lokalny interfejs sieciowy, który ma być używany, albo port lokalny, albo zakres portów lokalnych, które mają być używane.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : LOCALADDRESS

Krótką nazwą narzędzia administracyjnego JMS : LA

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setLocalAddress ()
- MQConnectionFactory.getLocalAdres ()

Wartości

"" (pusty łańcuch)

Jest to wartość domyślna.

Łańcuch w formacie [ip-addr] [(port niskotowy [, port])]

Poniżej przedstawiono kilka przykładów:

192.0.2.0

Kanał łączy się lokalnie z adresem 192.0.2.0 .

192.0.2.0(1000)

Kanał łączy się z adresem 192.0.2.0 lokalnie i używa portu 1000.

192.0.2.0(1000,2000)

Kanał łączy się lokalnie z adresem 192.0.2.0 i korzysta z portu w zakresie od 1000 do 2000.

(1000)

Kanał łączy się lokalnie z portem 1000.

(1000,2000)

Kanał łączy się lokalnie z portem w zakresie od 1000 do 2000.

Zamiast adresu IP można podać nazwę hosta. W przypadku połączenia w czasie rzeczywistym z brokerem ta właściwość ma znaczenie tylko wtedy, gdy używana jest funkcja rozsyłania grupowego, a wartość właściwości nie może zawierać numeru portu ani zakresu numerów portów. Jedynymi poprawnymi wartościami właściwości w tym przypadku są null, adres IP lub nazwa hosta.

MAPNAMESTYLE

Umożliwia użycie stylu zgodności dla nazw elementów MapMessage .

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : MAPNAMESTYLE

Krótką nazwa narzędzia administracyjnego JMS : MNST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Wartości

STANDARDOWA

Ma być używany standardowy format nazewnictwa elementów com.ibm.jms.JMSMapMessage . Jest to wartość domyślna, która umożliwia użycie nieprawnych identyfikatorów produktu Java jako nazwy elementu.

Kompatybilny

Ma być używany starszy format nazewnictwa elementów com.ibm.jms.JMSMapMessage . Jako nazwy elementu mogą być używane tylko legalne identyfikatory Java . Jest to potrzebne tylko wtedy, gdy komunikaty mapy są wysyłane do aplikacji korzystającej z wersji produktu IBM MQ classes for JMS wcześniejszej niż wersja 5.3.

MAXBUFFSIZE

Maksymalna liczba odebranych komunikatów, które mogą być zapisane w wewnętrznym buforze komunikatów podczas oczekiwania na przetworzenie przez aplikację. Ta właściwość ma zastosowanie tylko wtedy, gdy TRANSPORT ma wartość DIRECT lub DIRECTHTTP.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : MAXBUFFSIZE

Krótką nazwa narzędzia administracyjnego JMS : MBSZ

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Wartości

1000

Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

MDREAD

Ta właściwość określa, czy aplikacja produktu JMS może wyodrębnić wartości pól MQMD.

Obiekty mające zastosowanie

Długa nazwa narzędzia administracyjnego JMS : MDREAD

Krótką nazwa narzędzia administracyjnego JMS : MDR

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Wartości

NO

Podczas wysyłania komunikatów właściwości JMS_IBM_MQMD* wysłanego komunikatu nie są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD. Podczas odbierania komunikatów żadna właściwość JMS_IBM_MQMD* nie jest dostępna w odebranym komunikacie, nawet jeśli nadawca ustawił niektóre lub wszystkie z nich. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów użyj wartości False.

Tak

Podczas wysyłania komunikatów wszystkie właściwości JMS_IBM_MQMD* wysłanego komunikatu są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD, w tym właściwości, które nie zostały jawnie ustawione przez nadawcę. Podczas odbierania komunikatów wszystkie właściwości JMS_IBM_MQMD* są dostępne w odebranym komunikacie, łącznie z właściwościami, które nie zostały jawnie ustawione przez nadawcę.

W przypadku programów należy użyć wartości True.

MDWRITE

Ta właściwość określa, czy aplikacja produktu JMS może ustawiać wartości pól MQMD.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : MDWRITE

Krótką nazwa narzędzia administracyjnego JMS : MDR

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Wartości

NO

Wszystkie właściwości JMS_IBM_MQMD* są ignorowane, a ich wartości nie są kopiowane do bazowej struktury MQMD. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów użyj wartości False.

YES

Przetwarzane są właściwości JMS_IBM_MQMD*. Ich wartości są kopiowane do bazowej struktury MQMD.

W przypadku programów należy użyć wartości True.

MDMSGCTX

Jaki poziom kontekstu komunikatu ma zostać ustawiony przez aplikację JMS . Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu.

Obiekty mające zastosowanie

Długa nazwa narzędzia administracyjnego JMS : MDMSGCTX

Krótką nazwa narzędzia administracyjnego JMS : MDCTX

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Wartości

DEFAULT

Wywołanie funkcji API MQOPEN i struktura MQPMO nie określają jawnych opcji kontekstu komunikatu. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT,

Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_IDENTITY_CONTEXT, a struktura MQPMO określa wartość MQPMO_SET_IDENTITY_CONTEXT.

W przypadku programów należy użyć wartości WMQ_MDCTX_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT

Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_ALL_CONTEXT, a struktura MQPMO określa parametr MQPMO_SET_ALL_CONTEXT.

W przypadku programów należy użyć wartości WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Maksymalna liczba komunikatów, które mają być pobrane z kolejki w jednym pakiecie w przypadku używania asynchronicznego dostarczania komunikatów.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : MAXBUFFSIZE

Krótką nazwa narzędzia administracyjnego JMS : MBSZ

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Wartości

10

Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

MSGBODY

Określa, czy aplikacja JMS uzyskuje dostęp do MQRFH2 komunikatu IBM MQ jako część ładunku komunikatu.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : WMQ_MESSAGE_BODY

Krótką nazwa narzędzia administracyjnego JMS : MBODY

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Wartości

Nieokreślone

Podczas wysyłania produkt IBM MQ classes for JMS nie generuje lub nie zawiera nagłówka MQRFH2 , w zależności od wartości właściwości WMQ_TARGET_CLIENT. Podczas odbierania działa jako wartość JMS.

JMS

Po wysłaniu produkt IBM MQ classes for JMS automatycznie generuje nagłówek MQRFH2 i dołącza go do komunikatu IBM MQ .

Po odebraniu produkt IBM MQ classes for JMS ustawia właściwości komunikatu produktu JMS zgodnie z wartościami w tabeli MQRFH2 (jeśli jest obecny). Nie jest ona obecna w treści komunikatu MQRFH2 jako część treści komunikatu produktu JMS .

MQ

Podczas wysyłania program IBM MQ classes for JMS nie generuje MQRFH2.

Po odebraniu produkt IBM MQ classes for JMS przedstawia element MQRFH2 jako część treści komunikatu produktu JMS .

MSGRETENTION

Określa, czy konsument połączenia przechowuje niedostarczone komunikaty w kolejce wejściowej.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, XAConnectionFactory, XAQueueConnection,

Długa nazwa narzędzia administracyjnego JMS : MSGRETENTION

Krótką nazwa narzędzia administracyjnego JMS : MRET

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMessageRetention ()
- MQConnectionFactory.getMessageRetention ()

Wartości

Tak

Niedostarczone komunikaty pozostają w kolejce wejściowej. Jest to wartość domyślna.

Nie

Niedostarczone wiadomości są traktowane zgodnie z ich opcjami dyspozycji.

MSGSELECTION

Określa, czy wybór komunikatów jest dokonany przez IBM MQ classes for JMS, czy przez brokera. Jeśli TRANSPORT ma wartość DIRECT, wybór komunikatów jest zawsze przeprowadzany przez broker, a wartość parametru MSGSELECTION jest ignorowana. Wybór komunikatu przez broker nie jest obsługiwany, gdy BROKERVER ma wartość V1.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : MSGSELECTION

Krótką nazwa narzędzia administracyjnego JMS : MSEL

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMessageSelection ()
- MQConnectionFactory.getMessageWybór ()

Wartości

KLIENT

Wybór komunikatów jest dokonany przez produkt IBM MQ classes for JMS. Jest to wartość domyślna.

BROKER

Wybór komunikatu jest dokonany przez broker.

MULTICAST

Aby włączyć rozsyłanie grupowe w czasie rzeczywistym do brokera oraz, jeśli jest to możliwe, określić dokładny sposób, w jaki rozsyłanie jest używane do dostarczania komunikatów z brokera do konsumenta komunikatów. Właściwość nie ma wpływu na to, w jaki sposób producent komunikatów wysyła komunikaty do brokera.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, temat

Długa nazwa narzędzia administracyjnego JMS : MULTICAST

Krótką nazwa narzędzia administracyjnego JMS : MCAST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Wartości

WYŁĄCZONE

Komunikaty nie są dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego. Jest to wartość domyślna dla obiektów fabryki połączeń ConnectionFactory i TopicConnection.

ASCF

Komunikaty są dostarczane do konsumenta komunikatów zgodnie z ustawieniem rozsyłania grupowego dla fabryki połączeń powiązanej z konsumentem komunikatów. Ustawienie rozsyłania grupowego dla fabryki połączeń jest oznaczane w momencie tworzenia konsumenta komunikatów. Ta wartość jest poprawna tylko dla obiektów tematu i jest to wartość domyślna dla obiektów tematu.

WŁĄCZONY

Jeśli temat został skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego. Jeśli temat został skonfigurowany pod kątem niezawodnego rozsyłania grupowego, używana jest niezawodna jakość usługi.

Niezawodne

Jeśli temat został skonfigurowany pod kątem niezawodnego rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu transportu rozsyłania grupowego z niezawodną jakością usługi. Jeśli temat nie został skonfigurowany pod kątem niezawodnego rozsyłania grupowego, nie można utworzyć konsumenta komunikatów dla tego tematu.

NOTR

Jeśli temat został skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu rozsyłania grupowego. Niezawodna jakość usługi nie jest używana, nawet jeśli temat został skonfigurowany pod kątem niezawodnego rozsyłania grupowego.

OPTIMISTICPUBLICATION

Ta właściwość określa, czy produkt IBM MQ classes for JMS zwraca element sterujący natychmiast do publikatora, który opublikował komunikat, czy też zwraca element sterujący tylko po zakończeniu wszystkich przetwarzania powiązanych z wywołaniem i może zgłosić wynik do publikatora.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : OPTIMISTICPUBLICATION

Krótką nazwą narzędzia administracyjnego JMS : OPTPUB

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setOptimisticPublikacja ()
- MQConnectionFactory.getOptimisticPublikacja ()

Wartości

NO

Gdy publikator publikuje komunikat, produkt IBM MQ classes for JMS nie zwraca elementu sterującego do publikatora, dopóki nie zakończy przetwarzania związanego z wywołaniem i nie będzie mógł zgłosić wyniku do publikatora. Jest to wartość domyślna.

YES

Gdy publikator publikuje komunikat, program IBM MQ classes for JMS zwraca sterowanie do publikatora natychmiast, zanim zakończy przetwarzanie powiązane z wywołaniem i może zgłosić wynik do publikatora. Produkt IBM MQ classes for JMS zgłasza wynik tylko wtedy, gdy publikator zatwierdza komunikat.

OUTCOMENOTIFICATION

Ta właściwość określa, czy produkt IBM MQ classes for JMS zwraca kontrolę bezpośrednio do subskrybenta, który właśnie przyznał lub zatwierdził komunikat, czy też zwraca kontrolę dopiero po zakończeniu przetwarzania związanego z wywołaniem i może zgłosić wynik do subskrybenta.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : OUTCOMENOTIFICATION

Krótką nazwa narzędzia administracyjnego JMS : NOTIFY

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setOutcomePowiadomienie ()
- MQConnectionFactory.getOutcomePowiadomienie ()

Wartości

YES

Gdy subskrybent zatwierdzi lub zatwierdzi komunikat, program IBM MQ classes for JMS nie zwraca elementu sterującego do subskrybenta, dopóki nie zakończy przetwarzania związanego z wywołaniem i może zgłosić wynik do subskrybenta. Jest to wartość domyślna.

NO

Gdy subskrybent potwierdza lub zatwierdza komunikat, program IBM MQ classes for JMS zwraca kontrolę do subskrybenta natychmiast, zanim zakończy przetwarzanie powiązane z wywołaniem i może zgłosić wynik do subskrybenta.

PERSISTENCE

Trwałość komunikatów wystanych do miejsca docelowego.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : PERSISTENCE

Krótką nazwa narzędzia administracyjnego JMS : PER

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Wartości

APP

Trwałość jest definiowana przez aplikację JMS . Jest to wartość domyślna.

QDEF

Trwałość przyjmuje wartość domyślną kolejki.

PERS

Komunikaty są trwałe.

NIE

Komunikaty są nietrwałe.

WYSOKA

Więcej informacji na temat używania tej wartości można znaleźć w sekcji [Komunikaty trwałe produktuJMS](#).

POLLINGINT

Jeśli każdy obiekt nasłuchiwanie komunikatów w sesji nie ma odpowiedniego komunikatu w swojej kolejce, jest to maksymalny odstęp czasu (w milisekundach), jaki upływa przed ponowną próbą pobrania komunikatu z kolejki przez każdy obiekt nasłuchiwanie komunikatów. Jeśli często zdarza się, że żaden odpowiedni komunikat nie jest dostępny dla żadnego z obiektów nasłuchiwanie komunikatów w sesji, należy rozważyć zwiększenie wartości tej właściwości. Ta właściwość ma znaczenie tylko wtedy, gdy `TRANSPORT` ma wartość `BIND` lub `CLIENT`.

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `QueueConnection`, fabryka `TopicConnection`, fabryka `XAConnectionFactory`, fabryka `XAQueueConnection`, fabryka `XATopicConnection`.

Długa nazwa narzędzia administracyjnego JMS : `POLLINGINT`

Krótką nazwa narzędzia administracyjnego JMS : `PINT`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setPollingOdstęp czasu ()`
- `MQConnectionFactory.getPollingInterwał ()`

Wartości

5000

Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

PORT

W przypadku połączenia z menedżerem kolejek jest to numer portu, na którym nasłuchuje menedżer kolejek lub, w przypadku połączenia w czasie rzeczywistym z brokerem, numer portu, na którym broker nasłuchuje połączeń w czasie rzeczywistym.

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `QueueConnection`, fabryka `TopicConnection`, fabryka `XAConnectionFactory`, fabryka `XAQueueConnection`, fabryka `XATopicConnection`.

Długa nazwa narzędzia administracyjnego JMS : `PORT`

Krótką nazwa narzędzia administracyjnego JMS : `PORT`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setPort()`
- `MQConnectionFactory.getPort()`

Wartości

1414

Jest to wartość domyślna, jeśli TRANSPORT jest ustawiony na wartość CLIENT.

1506

Jest to wartość domyślna, jeśli TRANSPORT jest ustawiony na wartość DIRECT lub DIRECTHTTP.

Dowolna dodatnia liczba całkowita

PRIORYTET

Priorytet komunikatów wysyłanych do miejsca docelowego.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : PRIORITY

Krótką nazwa narzędzia administracyjnego JMS : PRI

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setPriority()
- MQDestination.getPriority()

Wartości

APP

Priorytet jest definiowany przez aplikację JMS . Jest to wartość domyślna.

QDEF

Priorytet przyjmuje wartość domyślną kolejki.

Dowolna liczba całkowita z zakresu od 0 do 9

Najniższy do najwyższego.

PROCESSDURATION

Ta właściwość określa, czy subskrybent gwarantuje szybkie przetwarzanie dowolnego komunikatu, który otrzymuje przed zwróceniem elementu sterującego do produktu IBM MQ classes for JMS.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROCESSDURATION

Krótką nazwa narzędzia administracyjnego JMS : PROCUR

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setProcessCzas trwania ()
- MQConnectionFactory.getProcessCzas trwania ()

Wartości

NIEZNANY

Subskrybent nie może udzielić gwarancji na to, jak szybko może przetworzyć otrzymany przez niego komunikat. Jest to wartość domyślna.

Krótki

Subskrybent gwarantuje szybkie przetwarzanie wszystkich komunikatów, które otrzymuje przed zwróceniem sterowania do produktu IBM MQ classes for JMS.

PROVIDERVERSION

Ta właściwość rozróżnia trzy tryby przesyłania komunikatów produktu IBM MQ : tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ , tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ z ograniczeniami oraz tryb migracji dostawcy przesyłania komunikatów produktu IBM MQ .

Tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ korzysta ze wszystkich funkcji menedżera kolejek produktu IBM MQ w celu zaimplementowania produktu JMS. Ten tryb jest zoptymalizowany pod kątem korzystania z funkcji API i funkcji API produktu JMS 2.0 . Tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ z ograniczeniami korzysta z interfejsu API produktu JMS 2.0 , ale nie korzysta z nowych funkcji, takich jak subskrypcje współużytkowane, opóźnione dostarczanie lub wysyłanie asynchroniczne.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection , fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : PROVIDERVERSION

Krótką nazwą narzędzia administracyjnego JMS : PVER

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setProviderWersja ()
- MQConnectionFactory.getProviderWersja ()

Wartości

W przypadku właściwości **PROVIDERVERSION** można ustawić dowolną wartość: 8 (tryb normalny), 7 (tryb normalny z ograniczeniami), 6 (tryb migracji) lub nieokreślona (wartość domyślna). Wartość podana we właściwości **PROVIDERVERSION** musi być łańcuchem. Jeśli jest podana opcja 8, 7 lub 6, dopuszczalne są następujące formaty:

- V.R.M.F
- V.R.M
- V.R
- V

Gdzie: V, R, M i F są wartościami całkowitymi większymi niż zero lub równymi zero. Dodatkowe wartości R, M i F są opcjonalne i można ich używać, jeśli wymagana jest precyzyjna kontrola. Aby na przykład użyć poziomu **PROVIDERVERSION** 7, w przypadku właściwości **PROVIDERVERSION** można podać wartość 7, 7.0, 7.0.0 lub 7.0.0.0.

8 – tryb normalny

Aplikacja JMS używa trybu normalnego dostawcy usługi przesyłania komunikatów produktu IBM MQ. W trybie normalnym używane są wszystkie funkcje menedżera kolejek produktu IBM MQ służące do implementowania usług JMS. Ten tryb jest zoptymalizowany pod kątem użycia funkcjonalności i interfejsu API JMS 2.0.

W przypadku nawiązywania połączenia z menedżerem kolejek przy poziomie komend 800 mogą być używane wszystkie interfejsy API JMS 2.0 i funkcje, takie jak wysyłanie asynchroniczne, opóźniona dostawa czy subskrypcja współużytkowana.

Jeśli menedżer kolejek podany w ustawieniach fabryki połączeń nie jest menedżerem kolejek w wersji 8.0.0, metoda `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSFMQ0003`.

Tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ korzysta z funkcji konwersacji współużytkowanych, a liczba konwersacji, które mogą być współużytkowane, jest kontrolowana przez właściwość **SHARECNV()** na kanale połączenia z serwerem. Jeśli w przypadku tej właściwości jest ustawiona wartość 0, nie można używać trybu normalnego dostawcy usługi przesyłania komunikatów produktu IBM MQ, a wykonywanie metody `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSCC5007`.

7 – tryb normalny z ograniczeniami

Aplikacja JMS używa trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ. W tym trybie używany jest interfejs API JMS 2.0, ale nie są używane nowe funkcje, takie jak subskrypcje współużytkowane, opóźniona dostawa czy wysyłanie asynchroniczne.

Jeśli w przypadku atrybutu **PROVIDERVERSION** zostanie ustawiona wartość 7, dostępny będzie tylko normalny tryb operacji z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ. Jeśli menedżer kolejek podany w ustawieniach fabryki połączeń nie jest menedżerem kolejek w wersji 7.0.1 lub nowszej, wykonywanie metody `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSFCC5008`.

W przypadku nawiązywania połączenia w trybie normalnym z ograniczeniami z menedżerem kolejek przy poziomie komend od 700 do 800, może być używany interfejs API JMS 2.0, ale bez wysyłania asynchronicznego, opóźnionej dostawy i subskrypcji współużytkowanej.

W trybie normalnym z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ używana jest funkcja **SHARECNV()** w kanale połączenia z serwerem. Jeśli w przypadku tej właściwości jest ustawiona wartość 0, nie można używać trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ, a wykonywanie metody `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSCC5007`.

6 – tryb migracji

Aplikacja JMS używa trybu migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

Aplikacja IBM MQ classes for JMS używa funkcji i algorytmów dostarczonych z produktem IBM WebSphere MQ 6.0. Aby nawiązać połączenie w produkcie WebSphere Message Broker 6.0 lub 6.1 przy użyciu produktu IBM WebSphere MQ Enterprise Transport 6.0, należy użyć tego trybu. Używając tego trybu, można nawiązać połączenie z menedżerem kolejek produktu IBM MQ 8.0, ale żadna nowa funkcja (np. odczyt z wyprzedzeniem lub przetwarzanie strumieniowe) menedżera kolejek IBM MQ classes for JMS nie będzie używana.

Jeśli klient produktu IBM MQ 8.0 nawiązuje połączenie z menedżerem kolejek produktu IBM MQ 8.0, wówczas wybór komunikatów jest dokonywany przez menedżer kolejek, a nie przez system klienta.

Jeśli określono tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ i zostanie podjęta próba użycia dowolnego interfejsu API JMS 2.0, wywołanie metody API zostanie zakończone niepowodzeniem i zostanie zgłoszony wyjątek `JMSCC5007`.

nieokreślona (wartość domyślna)

Domyślnie w przypadku właściwości **PROVIDERVERSION** ustawiona jest wartość *nieokreślona*.

Fabryka połączeń, która została utworzona w poprzedniej wersji produktu IBM MQ classes for JMS w interfejsie JNDI przejmuje tę wartość, gdy fabryka połączeń jest używana z nową wersją produktu IBM MQ classes for JMS. Do określania używanego trybu operacji używany jest poniższy algorytm. Ten algorytm jest używany w przypadku wywołania metody `createConnection`. Są w nim również używane inne aspekty fabryki połączeń w celu określenia, czy wymagany jest tryb normalny przesyłania komunikatów produktu IBM MQ, tryb normalny z ograniczeniami, czy tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

1. Najpierw podejmowana jest próba użycia dostawcy usługi przesyłania komunikatów produktu IBM MQ.
2. Jeśli połączony jest inny menedżer kolejek niż menedżer produktu IBM MQ 8.0 lub nowszego, podejmowana jest próba użycia trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ.
3. Jeśli połączony jest inny menedżer kolejek niż menedżer produktu IBM WebSphere MQ 7.0.1 lub nowszego, połączenie jest zamykane i w zamian używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
4. Jeśli w przypadku właściwości **SHARECNV** kanału połączenia z serwerem ustawiona jest wartość 0, połączenie jest zamykane i w zamian używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
5. Jeśli w przypadku właściwości **BROKERVER** ustawiona jest wartość V1 lub wartość domyślna *nieokreślona*, nadal używany jest tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ, co oznacza, że wszystkie operacje publikowania/subskrybowania używają nowych funkcji z produktu IBM WebSphere MQ 7.0.1 lub nowszego.

Sekcja [ALTER QMGR](#) zawiera informacje o parametrze PSMODE komendy ALTER QMGR, w tym dokładniejsze informacje o kompatybilności.

6. Jeśli w przypadku parametru **BROKERVER** ustawiona jest wartość V2, podejmowane działanie zależy od wartości parametru **BROKERQMGR**:
 - Jeśli w przypadku parametru **BROKERQMGR** ustawiono wartość blank:

Jeśli kolejka podana we właściwości **BROKERCONQ** może zostać otwarta dla danych wyjściowych (co jest równoznaczne z powodzeniem wykonania komendy MQOPEN dla danych wyjściowych), a w przypadku właściwości **PSMODE** w menedżerze kolejek ustawiona jest wartość COMPAT lub DISABLED, wówczas używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
 - Jeśli kolejka podana we właściwości **BROKERCONQ** nie może zostać otwarta dla danych wyjściowych lub jeśli w przypadku atrybutu **PSMODE** ustawiono wartość ENABLED:

Używany jest tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ.
 - Jeśli w przypadku właściwości **BROKERQMGR** podano wartość non-blank:

Używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

Jeśli nie można zmienić używanej fabryki połączeń, można użyć właściwości `com.ibm.msg.client.wmq.overrideProviderVersion`, aby nadpisać dowolne ustawienie w fabryce połączeń. To nadpisanie ma zastosowanie do wszystkich fabryk połączeń w maszynie JVM, ale rzeczywiste obiekty fabryki połączeń nie są modyfikowane.

Informacje pokrewne

Konfigurowanie właściwości JMS **PROVIDERVERSION**

[Reguły wybierania trybu dostawcy przesyłania komunikatów produktu IBM MQ](#)

PROXYHOSTNAME

Nazwa hosta lub adres IP systemu, na którym jest uruchomiony serwer proxy podczas korzystania z połączenia w czasie rzeczywistym z brokerem przez serwer proxy.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROXYHOSTNAME

Krótką nazwą narzędzia administracyjnego JMS : PHOST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Wartości

null

Nazwa hosta serwera proxy. Jest to wartość domyślna.

PROXYPORT

Numer portu, na którym nasłuchuje serwer proxy przy korzystaniu z połączenia w czasie rzeczywistym z brokerem przez serwer proxy.

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROXYPORT

Krótką nazwa narzędzia administracyjnego JMS : PPORT

Dostęp programistyczny

Procedury ustawiające/pobierające

MQConnectionFactory.setProxyPort ()

MQConnectionFactory.getProxyPort ()

Wartości

443

Numer portu serwera proxy. Jest to wartość domyślna.

PUBACKINT

Liczba komunikatów publikowanych przez publikatora przed żądaniem potwierdzenia od brokera przez produkt IBM MQ classes for JMS .

W przypadku obniżenia wartości tej właściwości produkt IBM MQ classes for JMS częściej zwraca się do potwierdzeń, dlatego wydajność publikatora zmniejsza się. Jeśli wartość zostanie podniesiona, program IBM MQ classes for JMS zajmie dłuższy czas, aby zgłosić wyjątek, jeśli broker nie powiedzie się. Ta właściwość ma znaczenie tylko wtedy, gdy TRANSPORT ma wartość BIND lub CLIENT.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : PROXYPORT

Krótką nazwa narzędzia administracyjnego JMS : PPORT

Dostęp programistyczny

Procedury ustawiające/pobierające

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Wartości

25

Dowolna dodatnia liczba całkowita może być wartością domyślną.

PUTASYNCAALLOWED

Ta właściwość określa, czy producenci komunikatów mogą używać asynchronicznych operacji put w celu wysyłania komunikatów do tego miejsca docelowego.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : PUTASYNCAALLOWED

Krótką nazwa narzędzia administracyjnego JMS : PAALD

Dostęp programistyczny

Procedury ustawiające/pobierające

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Wartości

AS_DEST

Określ, czy dozwolone są operacje put asynchroniczne, odwołując się do definicji kolejki lub tematu.
Jest to wartość domyślna.

AS_Q_DEF

Określ, czy dozwolone są operacje put asynchroniczne, odwołując się do definicji kolejki.

AS_TOPIC_DEF

Określ, czy dozwolone są operacje put asynchroniczne, odwołując się do definicji tematu.

NO

Asynchroniczne operacje put są niedozwolone.

YES

Dozwolone są asynchroniczne operacje put.

QMANAGER

Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.

Jeśli jednak aplikacja korzysta z tabeli definicji kanału klienta w celu nawiązania połączenia z menedżerem kolejek, należy zapoznać się z sekcji [Korzystanie z tabeli definicji kanału klienta przy użyciu produktu IBM MQ classes for JMS](#).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, kolejka, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : QMANAGER

Krótką nazwa narzędzia administracyjnego JMS : QMGR

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setQueueManager ()
- MQConnectionFactory.getQueueManager ()

Wartości

"" (pusty łańcuch)

Dowolny łańcuch może być wartością domyślną.

QUEUE

Nazwa miejsca docelowego kolejki produktu JMS . Jest ona zgodna z nazwą kolejki używanej przez menedżer kolejek.

Obiekty mające zastosowanie

Kolejka

Długa nazwa narzędzia administracyjnego JMS : QUEUE

Krótką nazwa narzędzia administracyjnego JMS : QU

Wartości

Dowolny łańcuch

Dowolna poprawna nazwa kolejki produktu IBM MQ .

Informacje pokrewne

[Reguły nazewnictwa obiektów IBM MQ >](#)

READAHEADALLOWED

Ta właściwość określa, czy konsumenci komunikatów i przeglądarki kolejek mogą korzystać z odczytu z wyprzedzeniem w celu uzyskania nietrwałych komunikatów z tego miejsca docelowego do buforu wewnętrznego przed ich odebraniem.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : READAHEADALLOWED

Krótką nazwa narzędzia administracyjnego JMS : RAALD

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Wartości

AS_DEST

Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki lub tematu. Jest to wartość domyślna w narzędziach administracyjnych.

Użyj komendy WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST w programach.

AS_Q_DEF

Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki.

Użyj komendy WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF w programach.

AS_TOPIC_DEF

Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji tematu.

Użyj komendy WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF w programach.

NO

Odczyt z wyprzedzeniem jest niedozwolony.

W programach należy użyć komendy WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED .

YES

Odczyt z wyprzedzeniem jest dozwolony.

W programach należy użyć komendy WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED .

READAHEADCLOSEPOLICY

W przypadku komunikatów dostarczanych do asynchronicznego programu nasłuchującego komunikatów, co dzieje się z komunikatami w wewnętrznym buforze odczytu z wyprzedzeniem, gdy konsument komunikatów jest zamknięty.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : READAHEADCLOSEPOLICY

Krótką nazwa narzędzia administracyjnego JMS : RACP

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Wartości

DELIVER_ALL

Wszystkie komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem są dostarczane do obiektu nasłuchiwanego komunikatów aplikacji przed zwróceniem. Jest to wartość domyślna w narzędziach administracyjnych.

Użyj komendy WMQConstants.WMQ_READ_AHEAD_DELIVERALL w programach.

DELIVER_CURRENT

Tylko bieżące wywołanie nasłuchiwanego komunikatów kończy się przed zwróceniem, potencjalnie pozostawiając komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem, które następnie są usuwane.

Użyj komendy WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT w programach.

RECEIVECCSID

Właściwość docelowa, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Wartość jest ignorowana, chyba że właściwość RECEIVECONVERSION jest ustawiona na wartość WMQ_RECEIVE_CONVERSION_QMGR .

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : RECEIVECCSID

Krótką nazwa narzędzia administracyjnego JMS : RCCS

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Wartości

WMQConstants.WMQ_RECEIVE_CC_SID_JVM_DEFAULT

0 -Użyj wirtualnej maszyny języka Java `Charset.defaultCharset`

1208

UTF-8

CCSID

Obstługiwany identyfikator kodowanego zestawu znaków.

RECEIVECONVERSION

Właściwość miejsca docelowego, która określa, czy menedżer kolejek ma wykonać konwersję danych.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : RECEIVECONVERSION

Krótką nazwa narzędzia administracyjnego JMS : RCNV

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Wartości

WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG

1 -wykonuje tylko konwersję danych na kliencie JMS . Wartość domyślna z poziomu do V7.0i z, włącznie z, 7.0.1.5.

WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR

2 -Przeprowadź konwersję danych w menedżerze kolejek przed wystaniem komunikatu do klienta. Wartość domyślna (i tylko) z V7.0 do V7.0.1.4 włącznie, z wyjątkiem sytuacji, gdy zastosowano poprawkę APAR IC72897 .

RECEIVEISOLATION

Ta właściwość określa, czy subskrybent może odbierać komunikaty, które nie zostały zatwierdzone w kolejce subskrybenta.

Obiekty mające zastosowanie

Fabryka `ConnectionFactory`, `TopicConnection`

Długa nazwa narzędzia administracyjnego JMS : RECEIVEISOLATION

Krótką nazwa narzędzia administracyjnego JMS : RCVISOL

Wartości

ZATWIERDZONA

Subskrybent otrzymuje tylko te komunikaty w kolejce subskrybenta, które zostały zatwierdzone. Jest to wartość domyślna w narzędziach administracyjnych.

Użyj komendy WMQConstants.WMQ_RCVISOL_COMMITTED w programach.

NIEZATWIERDZONA

Subskrybent może odbierać komunikaty, które nie zostały zatwierdzone w kolejce subskrybenta.

W programach należy użyć komendy WMQConstants.WMQ_RCVISOL_UNCOMMITTED.

RECEXIT

Identyfikuje wyjście odbierania kanału lub sekwencję wyjść odbierania, które mają zostać uruchomione w ramach dziedziczenia.

Aby program IBM MQ classes for JMS mógł znaleźć wyjścia odbierania, dodatkowa konfiguracja może być wymagana. Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klas produktu IBM MQ dla usługi JMS pod kątem używania wyjść kanału](#).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : RECEXIT

Krótką nazwą narzędzia administracyjnego JMS : RCX

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setReceiveWyjście ()
- MQConnectionFactory.getReceiveWyjście ()

Wartości

null

Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, w którym każdy element ma jedną z następujących wartości:

- Nazwa klasy, która implementuje interfejs WMQReceiveExit (w przypadku wyjścia odbierania kanału napisanego w produkcie Java).
- Łańcuch w formacie *libraryName(entryPointNazwa)* (w przypadku wyjścia odbierania kanału, które nie jest zapisywane w produkcie Java).

Jest to wartość domyślna.

RECEXITINIT

Dane użytkownika, które są przekazywane do wyjścia odbierania kanału podczas ich wywołania.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : RECEXITINIT

Krótką nazwą narzędzia administracyjnego JMS : RCXI

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Wartości

null

Łańcuch składający się z jednego lub większej liczby elementów danych użytkownika oddzielonych przecinkami. Jest to wartość domyślna.

REPLYTOSTYLE

Określa, w jaki sposób zostanie skonstruowane pole JMSReplyTo w odebranych komunikacie.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : REPLYTOSTYLE

Krótką nazwa narzędzia administracyjnego JMS : RTOST

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Wartości

DEFAULT

Odpowiada wartości MQMD.

RFH2

Użyj wartości podanej w nagłówku RFH2 . Jeśli wartość JMSReplyTo została ustawiona w aplikacji wysyłającej, należy użyć tej wartości.

MQMD

Użyj wartości podanej w tabeli MQMD. To zachowanie jest równoważne domyślnym działaniem produktu IBM WebSphere MQ 6.0.2.4 i 6.0.2.5.

Jeśli wartość JMSReplyTo ustawiona przez aplikację wysyłającą nie zawiera nazwy menedżera kolejek, odbierający menedżer kolejek wstawia własną nazwę w strukturze MQMD. Jeśli ten parametr zostanie ustawiony na wartość MQMD, używana kolejka odpowiedzi będzie używana w odbierającym menedżerze kolejek. Jeśli ten parametr zostanie ustawiony na wartość RFH2, używana kolejka odpowiedzi znajduje się w menedżerze kolejek określonym w parametrze RFH2 wysłanego komunikatu zgodnie z oryginalnie ustawionym przez aplikację wysyłającą.

Jeśli wartość JMSReplyTo ustawiona przez aplikację wysyłającą zawiera nazwę menedżera kolejek, wartość tego parametru jest nieistotna, ponieważ zarówno wartość MQMD, jak i RFH2 zawierają tę samą wartość.

RESCANINT

Gdy konsument komunikatów w domenie typu punkt z punktem korzysta z selektora komunikatów w celu wybrania komunikatów, które mają być odbierane, IBM MQ classes for JMS przeszuka kolejkę IBM MQ

pod kątem odpowiednich komunikatów w kolejności określonej przez atrybut `MsgDeliverySequence` kolejki.

Po znalezieniu odpowiedniego komunikatu przez program IBM MQ classes for JMS i dostaniu go do konsumenta, program IBM MQ classes for JMS znów wyszukuje następnego odpowiedniego komunikatu z bieżącej pozycji w kolejce. Program IBM MQ classes for JMS kontynuuje wyszukiwanie w kolejce w ten sposób do momentu osiągnięcia końca kolejki lub do momentu, gdy upłynie czas (w milisekundach) określony przez wartość tej właściwości. W każdym przypadku program IBM MQ classes for JMS powróć na początku kolejki, aby kontynuować wyszukiwanie, a następnie rozpoczyna się nowy przedział czasu.

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `QueueConnection`, `XAConnectionFactory`, `XAQueueConnection`.

Długa nazwa narzędzia administracyjnego JMS : `RESCANINT`

Krótką nazwa narzędzia administracyjnego JMS : `RINT`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setRescanInterval ()`
- `MQConnectionFactory.getRescanInterval ()`

Wartości

5000

Dowolna dodatnia liczba całkowita może być wartością domyślną.

SECEXIT

Identyfikuje wyjście zabezpieczeń kanału.

Aby program IBM MQ classes for JMS mógł zlokalizować wyjścia zabezpieczeń, może być wymagana dodatkowa konfiguracja. Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klas produktu IBM MQ dla usługi JMS pod kątem używania wyjść kanału](#).

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `QueueConnection`, fabryka `TopicConnection`, fabryka `XAConnectionFactory`, fabryka `XAQueueConnection`, fabryka `XATopicConnection`.

Długa nazwa narzędzia administracyjnego JMS : `SECEXIT`

Krótką nazwa narzędzia administracyjnego JMS : `SXC`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setSecurityWyjście ()`
- `MQConnectionFactory.getSecurityWyjście ()`

Wartości

null

Nazwa klasy implementujący interfejs `WMQSecurityExit` (w przypadku wyjścia zabezpieczeń kanału napisanego w produkcie Java).

łańcuch w formacie *libraryName(entryPointNazwa)* (w przypadku wyjścia zabezpieczeń kanału nie napisanego w programie Java).

SECEXITINIT

Dane użytkownika przekazywane do wyjścia zabezpieczeń kanału w momencie jego wywołania.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SECEXITINIT

Krótką nazwa narzędzia administracyjnego JMS : SCXI

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Wartości

null

Dowolny łańcuch może być wartością domyślną.

SENDCHECKCOUNT

Liczba wywołań wysyłania, które umożliwią między sprawdzaniem błędów put asynchronicznym, w ramach pojedynczej sesji JMS , która nie jest transmiana.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SENDCHECKCOUNT

Krótką nazwa narzędzia administracyjnego JMS : SCC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Wartości

null

Dowolny łańcuch może być wartością domyślną.

SENDEXIT

Identyfikuje wyjście wysyłania kanału lub sekwencję wyjść wysyłania, które mają zostać uruchomione w ramach dziedziczenia.

Aby program IBM MQ classes for JMS mógł zlokalizować wyjścia wysyłania, może być wymagana dodatkowa konfiguracja. Więcej informacji na ten temat zawiera sekcja [Konfigurowanie klas produktu IBM MQ dla usługi JMS pod kątem używania wyjść kanału](#).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SENDEXIT

Krótką nazwa narzędzia administracyjnego JMS : SDX

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSendWyjście ()
- MQConnectionFactory.getSendWyjście ()

Wartości

null

Dowolny łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, w którym każdy element ma jedną z następujących wartości:

- Nazwa klasy implementujący interfejs WMQSendExit (w przypadku wyjścia wysyłania kanału napisanego w produkcie Java).
- Łańcuch w formacie *libraryName(entryPointNazwa)* (dla wyjścia wysyłania kanału, które nie zostało zapisane w produkcie Java).
-

Jest to wartość domyślna.

SENDEXITINIT

Dane użytkownika przekazywane do wyjść wysyłania kanału w momencie ich wywołania.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SENDEXITINIT

Krótką nazwa narzędzia administracyjnego JMS : SDXI

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

Wartości

null

Dowolny łańcuch składający się z jednego lub większej liczby elementów danych użytkownika oddzielonych przecinkami może być wartością domyślną.

SHARECONVALLOWED

Ta właściwość określa, czy połączenie klienta może współużytkować swoje gniazdo z innymi połączeniami JMS najwyższego poziomu z tego samego procesu do tego samego menedżera kolejek, jeśli definicje kanałów są zgodne.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SHARECONVALLOWED

Krótką nazwa narzędzia administracyjnego JMS : SCALD

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Wartości

YES

Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć komendy WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES.

NO

Ta wartość dotyczy narzędzi administracyjnych.

W przypadku programów należy użyć komendy WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO.

SPARSESUBS

Steruje strategią pobierania komunikatów obiektu TopicSubscriber .

Obiekty mające zastosowanie

Fabryka ConnectionFactory, TopicConnection

Długa nazwa narzędzia administracyjnego JMS : SPARSESUBS

Krótką nazwa narzędzia administracyjnego JMS : SSUBS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSparseSubskrypcje ()
- MQConnectionFactory.getSparseSubskrypcje ()

Wartości

NO

Subskrypcje otrzymują częste pasujące komunikaty. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości false.

YES

Subskrypcje odbierają rzadko zgodne komunikaty. Ta wartość wymaga, aby kolejka subskrypcji mogła zostać otwarta do przeglądania.

W przypadku programów należy użyć wartości true.

SSLCIPHERSUITE

Zestaw CipherSuite , który ma być używany do połączenia SSL.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SSLCIPHERSUITE

Krótką nazwa narzędzia administracyjnego JMS : SCPHS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSSLCipherSuite ()
- MQConnectionFactory.getSSLCipherSuite ()

Wartości

null

Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja [Właściwości SSL obiektów produktu JMS](#).

SSLCRL

Serwery CRL, które sprawdzają, czy nie ma odwołania do certyfikatu SSL.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SSLCRL

Krótką nazwa narzędzia administracyjnego JMS : SCRL

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSSLCertSklepy ()
- MQConnectionFactory.getSSLCertSklepy ()

Wartości

null

Rozdzielona spacjami lista adresów URL LDAP. Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja [Właściwości SSL obiektów produktu JMS](#).

SSLFIPSREQUIRED

Ta właściwość określa, czy połączenie SSL musi używać pakietu CipherSuite , który jest obsługiwany przez dostawcę IBM Java JSSE FIPS (IBMJSEFIPS).

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SSLFIPSREQUIRED

Krótką nazwa narzędzia administracyjnego JMS : SFIPS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSSLFipsWymagane ()
- MQConnectionFactory.getSSLFipsWymagane ()

Wartości

NO

Połączenie SSL może korzystać z dowolnego zestawu CipherSuite , który nie jest obsługiwany przez dostawcę IBM Java JSSE FIPS (IBMJSSEFIPS).

Jest to wartość domyślna. W programach użyj wartości false.

YES

Połączenie SSL musi używać pakietu CipherSuite , który jest obsługiwany przez program IBMJSSEFIPS.

W programach należy użyć wartości true.

SSLPEERNAME

W przypadku protokołu SSL: szkielet *nazwy wyróżniającej* , który musi być zgodny z udostępnionym przez menedżer kolejek.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SSLPEERNAME

Krótką nazwa narzędzia administracyjnego JMS : SPEER

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSSLPeerNazwa ()
- MQConnectionFactory.getSSLPeerNazwa ()

Wartości

null

Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja [Właściwości SSL obiektów produktu JMS](#).

SSLRESETCOUNT

W przypadku protokołu SSL: łączna liczba bajtów wysłanych i odebranych przez połączenie, zanim klucz tajny używany do szyfrowania jest ponownie negocjowany.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SSLRESETCOUNT

Krótką nazwa narzędzia administracyjnego JMS : SRC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSSLResetLiczba ()
- MQConnectionFactory.getSSLResetLiczba ()

Wartości

0

Zero lub dowolna dodatnia liczba całkowita mniejsza lub równa 999, 999, 999. Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja [Właściwości SSL obiektów produktu JMS](#).

STATREFRESHINT

Przedział czasu (w milisekundach) między odświeżeniami długotrwałego wykonywania transakcji, które wykrywa, kiedy subskrybent utraci połączenie z menedżerem kolejek.

Ta właściwość ma znaczenie tylko wtedy, gdy parametr SUBSTORE ma wartość QUEUE.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : STATREFRESHINT

Krótką nazwa narzędzia administracyjnego JMS : SRI

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Wartości

60000

Dowolna dodatnia liczba całkowita może być wartością domyślną. Więcej informacji na ten temat zawiera sekcja [Właściwości SSL obiektów produktu JMS](#).

SUBSTORE

W przypadku, gdy produkt IBM MQ classes for JMS przechowuje dane trwałe dotyczące aktywnych subskrypcji.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SUBSTORE

Krótką nazwa narzędzia administracyjnego JMS : SS

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSubscriptionStore ()
- MQConnectionFactory.getSubscriptionStore ()

Wartości

BROKER

Aby przechowywać szczegóły subskrypcji, należy użyć składnicy subskrypcji opartych na brokerze. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć komendy WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRATE

Przesyłanie informacji o subskrypcji z subskrypcji opartej na kolejce do składnicy subskrypcji opartych na brokerach.

W przypadku programów należy użyć komendy WMQConstants.WMQ_SUBSTORE_MIGRATE.

QUEUE

Aby przechowywać szczegóły subskrypcji, należy użyć składnicy subskrypcji opartej na kolejce.

W przypadku programów należy użyć komendy WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPPOINTALLGETS

Ta właściwość określa, czy wszystkie pobrania mają być wykonywane w punkcie synchronizacji.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : SYNCPPOINTALLGETS

Krótką nazwa narzędzia administracyjnego JMS : SPAG

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Wartości

Nie

Jest to wartość domyślna.

Tak

TARGCLIENT

Ta właściwość określa, czy format IBM MQ RFH2 jest używany do wymiany informacji z aplikacjami docelowymi.

Obiekty mające zastosowanie

Kolejka, temat

Długa nazwa narzędzia administracyjnego JMS : TARGCLIENT

Krótką nazwa narzędzia administracyjnego JMS : TC

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Wartości

JMS

Celem komunikatu jest aplikacja JMS . Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć komendy WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ

Celem komunikatu jest aplikacja inna niż JMS IBM MQ .

W przypadku programów należy użyć komendy WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Ta właściwość określa, czy komunikat odpowiedzi, wysyłany do kolejki identyfikowanej przez pole nagłówka JMSReplyTo komunikatu przychodzącego, ma nagłówek MQRFH2 tylko wtedy, gdy komunikat przychodzący ma nagłówek MQRFH2 .

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, XAConnectionFactory, XAQueueConnection.

Długa nazwa narzędzia administracyjnego JMS : TARGCLIENTMATCHING

Krótką nazwa narzędzia administracyjnego JMS : TCM

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Wartości

YES

Jeśli komunikat przychodzący nie ma nagłówka MQRFH2 , właściwość TARGCLIENT obiektu Queue pochodzącego z pola nagłówka JMSReplyTo komunikatu jest wysyłana do produktu MQ. Jeśli komunikat ma nagłówek MQRFH2 , właściwość TARGCLIENT jest zamiast tego ustawiona na wartość JMS . Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości true.

NO

Właściwość TARGCLIENT obiektu Queue pochodzącego z pola nagłówka JMSReplyTo komunikatu przychodzącego jest zawsze ustawiona na wartość JMS.

W przypadku programów należy użyć wartości false.

TEMPMODEL

Nazwa kolejki modelowej, z której tworzone są tymczasowe kolejki produktu JMS .

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, XAConnectionFactory, XAQueueConnection.

Długa nazwa narzędzia administracyjnego JMS : TEMPMODEL

Krótką nazwa narzędzia administracyjnego JMS : TM

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setTemporaryModel ()
- MQConnectionFactory.getTemporaryModel ()

Wartości

SYSTEM.DEFAULT.MODEL.QUEUE

Dowolny łańcuch może być wartością domyślną.

TEMPQPREFIX

Przedrostek używany do tworzenia nazwy kolejki dynamicznej IBM MQ .

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, XAConnectionFactory, XAQueueConnection.

Długa nazwa narzędzia administracyjnego JMS : TEMPQPREFIX

Krótką nazwa narzędzia administracyjnego JMS : TQP

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory.getTempQPrefix ()

Wartości

" " (pusty łańcuch)

The prefix used is CSQ . * on z/OS and AMQ . * on all other platforms. Są to wartości domyślne.

Przedrostek kolejki

Przedrostek kolejki to dowolny łańcuch, który jest zgodny z regułami tworzenia treści pola *DynamicQName* w deskrytorze obiektu IBM MQ (struktura MQOD), ale ostatni niepusty znak musi być gwiazdką.

TEMPTOPICPREFIX

Podczas tworzenia tematów tymczasowych program JMS generuje łańcuch tematu w postaci "TEMP / TEMPTOPICPREFIX/unique_id " lub jeśli ta właściwość zostanie pozostawiona z wartością domyślną, po prostu "TEMP /unique_id ". Określenie nie pustego elementu TEMPTOPICPREFIX umożliwia zdefiniowanie konkretnych kolejek modelowych na potrzeby tworzenia kolejek zarządzanych dla subskrybentów tematów tymczasowych utworzonych w ramach tego połączenia.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, TopicConnection, XAConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : TEMPTOPICPREFIX

Krótką nazwa narzędzia administracyjnego JMS : TTP

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Wartości

Dowolny łańcuch inny niż NULL składający się tylko z poprawnych znaków dla łańcucha tematu IBM MQ .
Wartością domyślną jest "" (pusty łańcuch).

TOPIC

Nazwa miejsca docelowego tematu produktu JMS , ta wartość jest używana przez menedżer kolejek jako łańcuch tematu publikacji lub subskrypcji.

Obiekty mające zastosowanie

Temat

Długa nazwa narzędzia administracyjnego JMS : TOPIC

Krótką nazwa narzędzia administracyjnego JMS : TOP

Wartości

Dowolny łańcuch

Łańcuch, który tworzy poprawny łańcuch tematu IBM MQ . Jeśli produkt IBM MQ jest używany jako dostawca przesyłania komunikatów z produktem WebSphere Application Server, należy określić wartość zgodną z nazwą, o której temat jest znany w celach administracyjnych w produkcie WebSphere Application Server.

Informacje pokrewne

[Łańcuchy tematów](#)

TRANSPORT

Rodzaj połączenia z menedżerem kolejek lub brokerem.

Obiekty mające zastosowanie

Fabryka połączeń ConnectionFactory, QueueConnection, fabryka TopicConnection, fabryka XAConnectionFactory, fabryka XAQueueConnection, fabryka XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : TRANSPORT

Krótką nazwa narzędzia administracyjnego JMS : TRAN

Dostęp programistyczny

Procedury ustawiające/pobierające

- MQConnectionFactory.setTransportTyp ()
- MQConnectionFactory.getTransportTyp ()

Wartości

BIND

W przypadku połączenia z menedżerem kolejek w trybie powiązań. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć komendy WMQConstants.WMQ_CM_BINDINGS.

KLIENT

W przypadku połączenia z menedżerem kolejek w trybie klienta.

W przypadku programów należy użyć komendy `WMQConstants.WMQ_CM_CLIENT`.

Bezpośrednia

W przypadku połączenia w czasie rzeczywistym z brokerem, który nie używa tunelowania HTTP.

W przypadku programów należy użyć komendy `WMQConstants.WMQ_CM_DIRECT_TCPIP`.

DIRECTHTTP

W przypadku połączenia w czasie rzeczywistym z brokerem przy użyciu tunelowania HTTP. Obsługiwany jest tylko protokół HTTP 1.0 .

W przypadku programów należy użyć komendy `WMQConstants.WMQ_CM_DIRECT_HTTP`.

Pojęcia pokrewne

[“Zależności między właściwościami obiektów produktu IBM MQ classes for JMS” na stronie 2036](#)
Poprawność niektórych właściwości jest zależna od konkretnych wartości innych właściwości.

WILDCARDFORMAT

Ta właściwość określa, która wersja składni ze znakami wieloznacznymi ma być używana.

Obiekty mające zastosowanie

Fabryka połączeń `ConnectionFactory`, `TopicConnection`, `XAConnectionFactory`, `XATopicConnection`.

Długa nazwa narzędzia administracyjnego JMS : `WILDCARDFORMAT`

Krótką nazwą narzędzia administracyjnego JMS : `WCFMT`

Dostęp programistyczny

Procedury ustawiające/pobierające

- `MQConnectionFactory.setWildCardFormat()`
- `MQConnectionFactory.getWildCardFormat()`

Wartości

TOPIC_ONLY

Rozpoznaje tylko znaki zastępcze poziomu tematu, które są używane w brokerze w wersji 2. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć komendy `WMQConstants.WMQ_WILDCARD_TOPIC_ONLY`.

CHAR_ONLY

Rozpoznaje tylko znaki wieloznaczne, które są używane w brokerze 1.

W przypadku programów należy użyć komendy `WMQConstants.WMQ_WILDCARD_CHAR_ONLY`.

Właściwość ENCODING

Właściwość `ENCODING` składa się z trzech podwłaściwości w dwunastu możliwych kombinacjach.

Poprawne wartości, jakie może wykonać właściwość `ENCODING` , są skonstruowane z trzech podwłaściwości:

Kodowanie na podstawie liczb całkowitych

Normalny lub odwrócony

Kodowanie dziesiętne

Normalny lub odwrócony

kodowanie zmiennopozycyjne

IEEE normal, IEEE reversed, or z/OS

Właściwość `ENCODING` jest wyrażona w postaci łańcucha o długości trzech znaków, z następującą składnią:

```
{N|R}{N|R}{N|R|3}
```

W tym łańcuchu:

- N oznacza normalny
- R oznacza odwrócone
- 3 oznacza z/OS
- Pierwszy znak reprezentuje *kodowanie liczb całkowitych*.
- Drugi znak reprezentuje *kodowanie dziesiętne*.
- Trzeci znak reprezentuje *kodowanie zmiennopozycyjne*.

Udostępnia zestaw dwunastu możliwych wartości dla właściwości `ENCODING`.

Istnieje dodatkowa wartość, łańcuch `NATIVE`, który ustawia odpowiednie wartości kodowania dla platformy Java.

W poniższych przykładach przedstawiono poprawne kombinacje dla produktu `ENCODING`:

```
ENCODING(NNR)  
ENCODING(NATIVE)  
ENCODING(RR3)
```

Właściwości SSL obiektów JMS

Włącz szyfrowanie SSL (Secure Sockets Layer) przy użyciu właściwości `SSLCIPHERSUITE`. Następnie można zmienić parametry szyfrowania SSL, korzystając z kilku innych właściwości.

Po określeniu wartości `TRANSPORT (CLIENT)` można włączyć komunikację szyfrowaną SSL (Secure Sockets Layer) przy użyciu właściwości `SSLCIPHERSUITE`. Tę właściwość należy ustawić na wartość `CipherSuite` udostępnianą przez dostawcę JSSE. Musi ona być zgodna z atrybutem `CipherSpec` o nazwie określonej w kanale `SVRCONN` o nazwie określonej przez właściwość `CHANNEL`.

Jednak atrybuty `CipherSpecs` (określone w kanale `SVRCONN`) i `CipherSuites` (określone w obiektach `ConnectionFactory`) używają różnych schematów nazewnictwa do reprezentowania tych samych algorytmów szyfrowania SSL. Jeśli w właściwości `SSLCIPHERSUITE` zostanie określona uznana nazwa `CipherSpec`, to `JMSAdmin` wysyła ostrzeżenie i odwzorowuje wartość `CipherSpec` na równoważną wartość `CipherSuite`. Listę `CipherSpecs` rozpoznawanych przez produkt IBM MQ i `JMSAdmin` można znaleźć w sekcji [SSL/TLS CipherSpecs i CipherSuites w produkcie IBM MQ classes for JMS](#).

Jeśli wymagane jest połączenie z użyciem pakietu `CipherSuite` obsługiwanego przez dostawcę FIPS IBM Java JSSE (`IBMJSSEFIPS`), ustaw właściwość `SSLFIPSREQUIRED` fabryki połączeń na `YES`. Wartością domyślną tej właściwości jest `NO`, co oznacza, że połączenie może korzystać z dowolnego obsługiwanego pakietu `CipherSuite`. Jeśli właściwość `SSLCIPHERSUITE` nie jest ustawiona, właściwość jest ignorowana.

Parametr `SSLPEERNAME` jest zgodny z formatem parametru `SSLPEER`, który można ustawić w definicjach kanałów. Jest to lista par nazw atrybutów i wartości rozdzielonych przecinkami lub średnikami. Na przykład:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPPHERE)
```

Zestaw nazw i wartości składa się z *nazwy wyróżniającej*. Więcej informacji na temat nazw wyróżniających i ich użycia z produktem IBM MQ zawiera sekcja [Bezpieczeństwo](#).

Podany przykład umożliwia sprawdzenie certyfikatu identyfikującego prezentowanego przez serwer w czasie połączenia. Aby nawiązanie połączenia powiodło się, certyfikat musi mieć nazwę `Common Name` rozpoczynającą się od `QMGR.`, i musi mieć co najmniej dwie nazwy jednostek organizacyjnych, z których pierwsza to `IBM`, a druga `WEBSPPHERE`. Podczas sprawdzania nie jest rozróżniana wielkość liter.

Jeśli parametr SSLPEERNAME nie jest ustawiony, sprawdzanie nie jest wykonywane. Parametr SSLPEERNAME jest ignorowany, jeśli parametr SSLCIPHERSUITE nie jest ustawiony.

Właściwość SSLCRL określa zero lub więcej serwerów CRL (Lista odwołań certyfikatów). Korzystanie z tej właściwości wymaga maszyny JVM w wersji Java 2 v1.4. Jest to rozdzielana spacjami lista wpisów w formularzu:

```
ldap:// hostname:[ port ]
```

opcjonalnie, po której następuje jeden/. Jeśli parametr *port* zostanie pominięty, przyjmowany jest domyślny port LDAP dla 389. W czasie połączenia certyfikat SSL przedstawiony przez serwer jest sprawdzany na podstawie określonych serwerów CRL. Więcej informacji na temat zabezpieczeń CRL zawiera sekcja [Bezpieczeństwo](#).

Jeśli opcja SSLCRL nie jest ustawiona, sprawdzanie nie jest wykonywane. Wartość SSLCRL jest ignorowana, jeśli parametr SSLCIPHERSUITE nie jest ustawiony.

Właściwość SSLRESETCOUNT reprezentuje łączną liczbę bajtów wysłanych i odebranych przez połączenie, zanim klucz tajny używany do szyfrowania jest ponownie negocjowany. Liczba wysłanych bajtów jest liczbą przed zaszyfrowaniem, a liczba odebranych bajtów jest liczbą po deszyfrowaniu. Liczba bajtów obejmuje również informacje sterujące wysłane i odebrane przez program IBM MQ classes for JMS.

Na przykład, aby skonfigurować obiekt ConnectionFactory, którego można użyć do utworzenia połączenia przez kanał MQI z włączoną obsługą SSL przy użyciu klucza tajnego, który jest ponownie negocjowany po 4 MB danych, należy wprowadzić następującą komendę do obiektu JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Jeśli wartość atrybutu SSLRESETCOUNT wynosi zero, co jest wartością domyślną, klucz tajny nigdy nie będzie ponownie negocjowany. Właściwość SSLRESETCOUNT jest ignorowana, jeśli właściwość SSLCIPHERSUITE nie jest ustawiona.

Niniejsza publikacja została opracowana z myślą o produktach i usługach oferowanych w Stanach Zjednoczonych.

IBM może nie oferować w innych krajach produktów, usług lub opcji omawianych w tej publikacji. Informacje o produktach i usługach dostępnych w danym kraju można uzyskać od lokalnego przedstawiciela IBM. Odwołanie do produktu, programu lub usługi IBM nie oznacza, że można użyć wyłącznie tego produktu, programu lub usługi IBM. Zamiast nich można zastosować ich odpowiednik funkcjonalny pod warunkiem, że nie narusza to praw własności intelektualnej firmy IBM. Jednakże cała odpowiedzialność za ocenę przydatności i sprawdzenie działania produktu, programu lub usługi pochodzących od producenta innego niż IBM spoczywa na użytkowniku.

IBM może posiadać patenty lub złożone wnioski patentowe na towary i usługi, o których mowa w niniejszej publikacji. Używanie tego dokumentu nie daje żadnych praw do tych patentów. Pisemne zapytania w sprawie licencji można przysyłać na adres:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Zapytania w sprawie licencji dotyczących informacji kodowanych przy użyciu dwubajtowych zestawów znaków (DBCS) należy kierować do lokalnych działów IBM Intellectual Property Department lub zgłaszać na piśmie pod adresem:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Poniższy akapit nie obowiązuje w Wielkiej Brytanii, a także w innych krajach, w których jego treść pozostaje w sprzeczności z przepisami prawa miejscowego: INTERNATIONAL BUSINESS MACHINES CORPORATION DOSTARCZA TĘ PUBLIKACJĘ W STANIE, W JAKIM SIĘ ZNAJDUJE ("AS IS"), BEZ JAKICHKOLWIEK GWARANCJI (RĘKOJMIĘ RÓWNIEŻ WYŁĄCZA SIĘ), WYRAŻNYCH LUB DOMNIEMANYCH, A W SZCZEGÓLNOŚCI DOMNIEMANYCH GWARANCJI PRZYDATNOŚCI HANDLOWEJ, PRZYDATNOŚCI DO OKREŚLONEGO CELU ORAZ GWARANCJI, ŻE PUBLIKACJA TA NIE NARUSZA PRAW OSÓB TRZECICH. Ustawodawstwa niektórych krajów nie dopuszczają zastrzeżeń dotyczących gwarancji wyraźnych lub domniemanych w odniesieniu do pewnych transakcji; w takiej sytuacji powyższe zdanie nie ma zastosowania.

Informacje zawarte w niniejszej publikacji mogą zawierać nieścisłości techniczne lub błędy typograficzne. Informacje te są okresowo aktualizowane, a zmiany te zostaną uwzględnione w kolejnych wydaniach tej publikacji. IBM zastrzega sobie prawo do wprowadzania ulepszeń i/lub zmian w produktach i/lub programach opisanych w tej publikacji w dowolnym czasie, bez wcześniejszego powiadomienia.

Wszelkie wzmianki w tej publikacji na temat stron internetowych innych podmiotów zostały wprowadzone wyłącznie dla wygody użytkowników i w żadnym wypadku nie stanowią zachęty do ich odwiedzania. Materiały dostępne na tych stronach nie są częścią materiałów opracowanych dla tego produktu IBM, a użytkownik korzysta z nich na własną odpowiedzialność.

IBM ma prawo do używania i rozpowszechniania informacji przystanych przez użytkownika w dowolny sposób, jaki uzna za właściwy, bez żadnych zobowiązań wobec ich autora.

Licencjodawcy tego programu, którzy chcieliby uzyskać informacje na temat programu w celu: (i) wdrożenia wymiany informacji między niezależnie utworzonymi programami i innymi programami (łącznie

z tym opisywanym) oraz (ii) wspólnego wykorzystywania wymienianych informacji, powinni skontaktować się z:

IBM Corporation
Koordynator współdziałania z oprogramowaniem, Dział 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Informacje takie mogą być udostępnione, o ile spełnione zostaną odpowiednie warunki, w tym, w niektórych przypadkach, zostanie uiszczona stosowna opłata.

Licencjonowany program opisany w niniejszej publikacji oraz wszystkie inne licencjonowane materiały dostępne dla tego programu są dostarczane przez IBM na warunkach określonych w Umowie IBM z Klientem, Międzynarodowej Umowie Licencyjnej IBM na Program lub w innych podobnych umowach zawartych między IBM i użytkownikami.

Wszelkie dane dotyczące wydajności zostały zebrane w kontrolowanym środowisku. W związku z tym rezultaty uzyskane w innych środowiskach operacyjnych mogą się znacząco różnić. Niektóre pomiary mogły być dokonywane na systemach będących w fazie rozwoju i nie ma gwarancji, że pomiary wykonane na ogólnie dostępnych systemach dadzą takie same wyniki. Niektóre z pomiarów mogły być estymowane przez ekstrapolację. Rzeczywiste wyniki mogą być inne. Użytkownicy powinni we własnym zakresie sprawdzić odpowiednie dane dla ich środowiska.

Informacje dotyczące produktów innych niż produkty IBM pochodzą od dostawców tych produktów, z opublikowanych przez nich zapowiedzi lub innych powszechnie dostępnych źródeł. Firma IBM nie testowała tych produktów i nie może potwierdzić dokładności pomiarów wydajności, kompatybilności ani żadnych innych danych związanych z tymi produktami. Pytania dotyczące możliwości produktów innych podmiotów należy kierować do dostawców tych produktów.

Wszelkie stwierdzenia dotyczące przyszłych kierunków rozwoju i zamierzeń IBM mogą zostać zmienione lub wycofane bez powiadomienia.

Publikacja ta zawiera przykładowe dane i raporty używane w codziennych operacjach działalności gospodarczej. W celu kompleksowego ich zilustrowania podane przykłady zawierają nazwiska osób prywatnych, nazwy przedsiębiorstw oraz nazwy produktów. Wszystkie te nazwy/nazwiska są fikcyjne i jakiegokolwiek podobieństwo do istniejących nazw/nazwisk i adresów jest całkowicie przypadkowe.

LICENCJA W ZAKRESIE PRAW AUTORSKICH:

Niniejsza publikacja zawiera przykładowe aplikacje w kodzie źródłowym, ilustrujące techniki programowania w różnych systemach operacyjnych. Użytkownik może kopiować, modyfikować i dystrybuować te programy przykładowe w dowolnej formie bez uiszczania opłat na rzecz IBM, w celu projektowania, używania, sprzedaży lub dystrybucji aplikacji zgodnych z aplikacyjnym interfejsem programistycznym dla tego systemu operacyjnego, dla którego napisane zostały programy przykładowe. Programy przykładowe nie zostały gruntownie przetestowane. IBM nie może zatem gwarantować ani sugerować niezawodności, użyteczności i funkcjonalności tych programów.

W przypadku przeglądania niniejszych informacji w formie elektronicznej, zdjęcia i kolorowe ilustracje mogą nie być wyświetlane.

Informacje dotyczące interfejsu programistycznego

Informacje dotyczące interfejsu programistycznego, o ile są udostępniane, mają być pomocne podczas tworzenia oprogramowania aplikacji do użytku z tym programem.

Ten podręcznik zawiera informacje na temat planowanych interfejsów programistycznych, które umożliwiają klientom pisanie programów w celu uzyskania dostępu do usług produktu WebSphere MQ.

Informacje te mogą również zawierać informacje na temat diagnostyki, modyfikacji i strojenia. Tego typu informacje są udostępniane jako pomoc przy debugowaniu aplikacji.

Ważne: Informacji na temat diagnostyki, modyfikacji i strojenia nie należy używać jako interfejsu programistycznego, ponieważ może on ulec zmianie.

Znaki towarowe

IBM, logo IBM, ibm.com, są znakami towarowymi IBM Corporation, zarejestrowanymi w wielu systemach prawnych na całym świecie. Aktualna lista znaków towarowych IBM jest dostępna w serwisie WWW, w sekcji "Copyright and trademark information" (Informacje o prawach autorskich i znakach towarowych), pod adresem www.ibm.com/legal/copytrade.shtml. Nazwy innych produktów lub usług mogą być znakami towarowymi IBM lub innych podmiotów.

Microsoft oraz Windows są znakami towarowymi Microsoft Corporation w Stanach Zjednoczonych i/lub w innych krajach.

UNIX jest zastrzeżonym znakiem towarowym The Open Group w Stanach Zjednoczonych i/lub w innych krajach.

Linux jest zastrzeżonym znakiem towarowym Linusa Torvaldsa w Stanach Zjednoczonych i/lub w innych krajach.

Ten produkt zawiera oprogramowanie opracowane przez Eclipse Project (<http://www.eclipse.org/>).

Java oraz wszystkie znaki towarowe i logo dotyczące języka Java są znakami towarowymi lub zastrzeżonymi znakami towarowymi Oracle i/lub przedsiębiorstw afiliowanych Oracle.



Numer pozycji:

(1P) P/N: